



US010489936B1

(12) **United States Patent**
Zafar et al.

(10) **Patent No.:** **US 10,489,936 B1**
(45) **Date of Patent:** **Nov. 26, 2019**

(54) **SYSTEM AND METHOD FOR LOSSY IMAGE AND VIDEO COMPRESSION UTILIZING A METANETWORK**

(71) Applicant: **Deep Render Ltd.**, London (GB)

(72) Inventors: **Arsalan Ali Zafar**, London (GB);
Christian Lars Besenbruch, London (GB)

(73) Assignee: **Deep Render Ltd.**, London (GB)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **16/413,770**

(22) Filed: **May 16, 2019**

Related U.S. Application Data

(63) Continuation-in-part of application No. 16/397,725, filed on Apr. 29, 2019, now Pat. No. 10,373,300.

(51) **Int. Cl.**

G06K 9/00 (2006.01)
G06T 9/00 (2006.01)
G06T 3/40 (2006.01)
G06N 20/20 (2019.01)
G06N 3/08 (2006.01)
G06N 3/04 (2006.01)
G06T 5/20 (2006.01)

(52) **U.S. Cl.**

CPC **G06T 9/002** (2013.01); **G06N 3/0454** (2013.01); **G06N 3/08** (2013.01); **G06N 20/20** (2019.01); **G06T 3/4046** (2013.01); **G06T 3/4084** (2013.01); **G06T 5/20** (2013.01)

(58) **Field of Classification Search**

None
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

7,647,618	B1 *	1/2010	Hunter	G06Q 20/123 709/229
2004/0045030	A1 *	3/2004	Reynolds	H04L 29/06 725/110
2005/0089215	A1 *	4/2005	Staelin	H04N 19/86 382/157
2006/0053086	A1 *	3/2006	Fazzini	H04N 19/172 706/15
2006/0212892	A1 *	9/2006	Hunter	H04N 5/913 725/1
2014/0072242	A1 *	3/2014	Wei	H04N 19/176 382/299
2016/0379352	A1 *	12/2016	Zhang	G06T 7/0002 382/157
2017/0230675	A1	8/2017	Wierstra et al.		
2017/0337682	A1 *	11/2017	Liao	G06T 7/30
2018/0150947	A1 *	5/2018	Lu	G06N 3/0454
2018/0174052	A1	6/2018	Rippel et al.		
2019/0087726	A1 *	3/2019	Greenblatt	G06N 3/08
2019/0138838	A1 *	5/2019	Liu	G06K 9/40
2019/0139191	A1 *	5/2019	Liu	G06T 5/10
2019/0139193	A1 *	5/2019	Navarrete Michelini	G06T 3/4069
2019/0206091	A1 *	7/2019	Weng	G06T 9/002

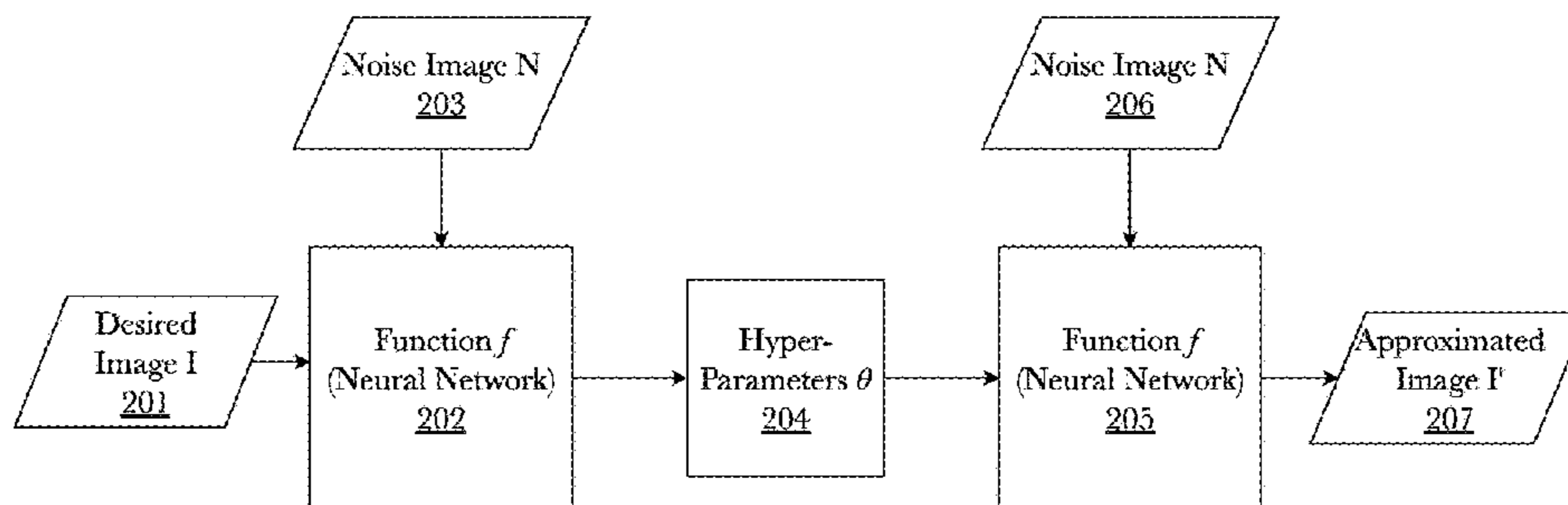
* cited by examiner

Primary Examiner — Shervin K Nakhjavan
(74) *Attorney, Agent, or Firm* — Brian R. Galvin; Galvin Patent Law, LLC

(57) **ABSTRACT**

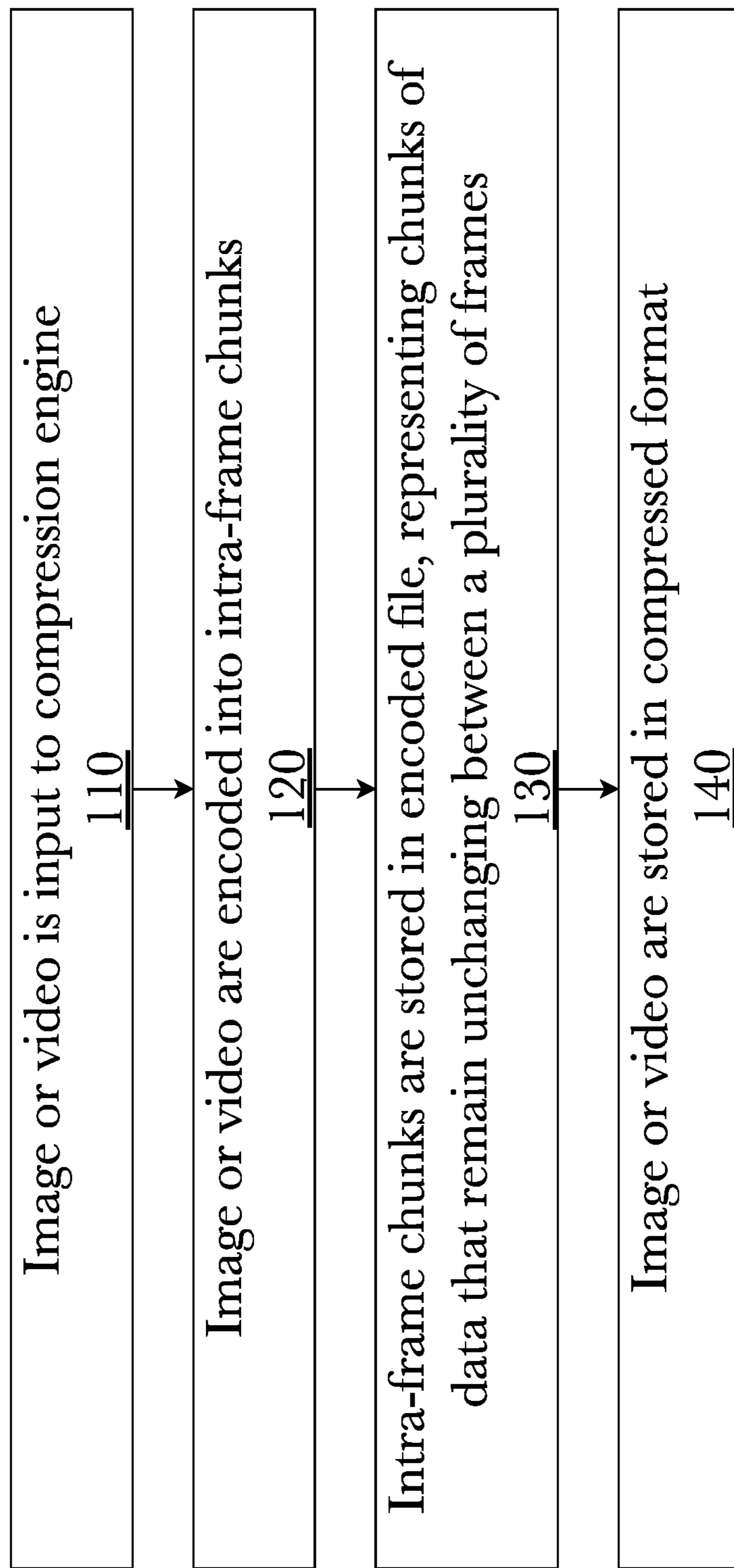
A system and method for lossy image and video compression that utilizes a metanetwork to generate a set of hyper-parameters necessary for an image encoding network to reconstruct the desired image from a given noise image.

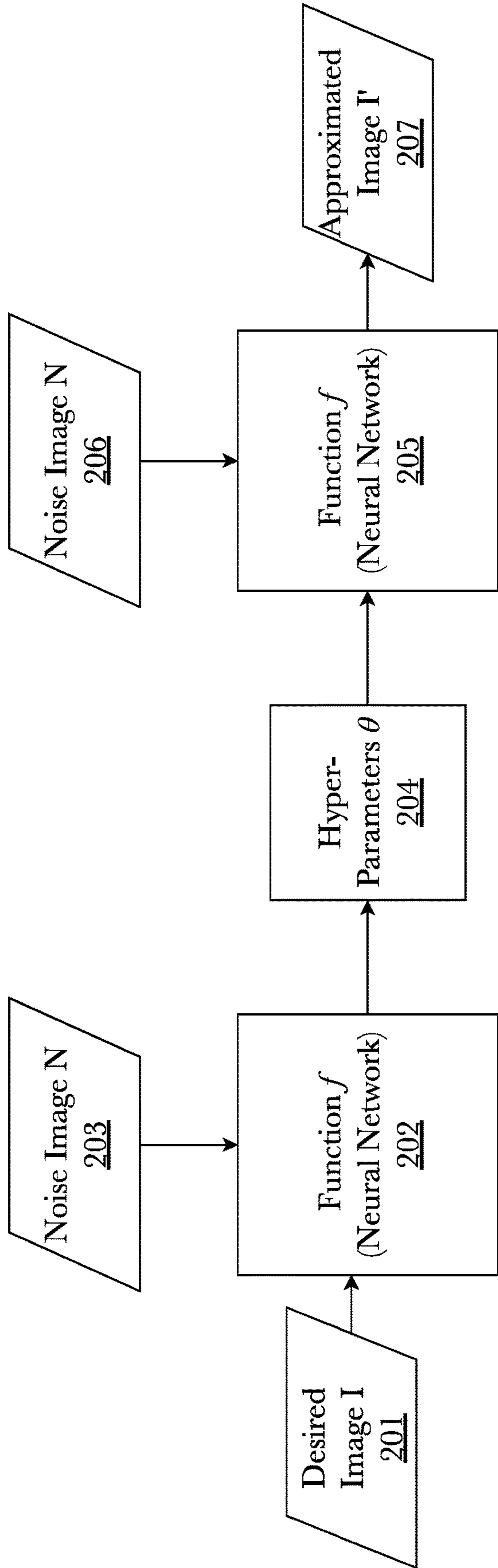
8 Claims, 24 Drawing Sheets



Given: I, N, ϵ
 Find: θ
 Such that: $f: \mathbb{R}^{W_N \times H_N \times C_N} \rightarrow \mathbb{R}^{I_N \times J_N \times K_N}$
 $f(\theta | N) := I'$
 $h(I', I) \leq \epsilon$

220

**Fig. 1 (PRIOR ART)**



Given: I, \mathcal{N}, ϵ
Find: θ
Such that: $f: \mathbb{R}^{W_N \cdot H_N \cdot C_N} \rightarrow \mathbb{R}^{I_N \cdot I_N \cdot I_N}$
 $f(\theta | \mathcal{N}) := I'$
 $h(I', I) \leq \epsilon$

220

Fig. 2

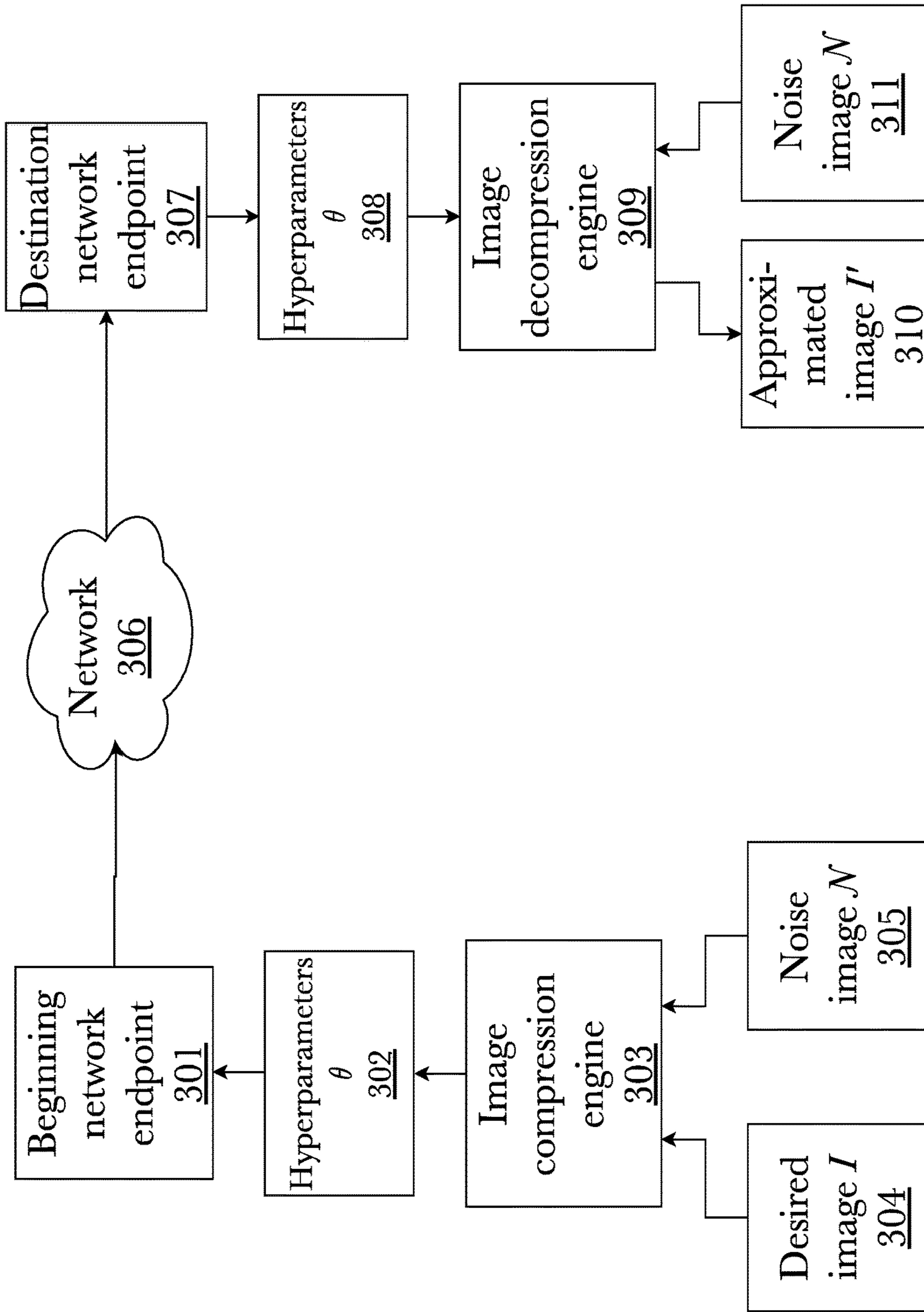


Fig. 3

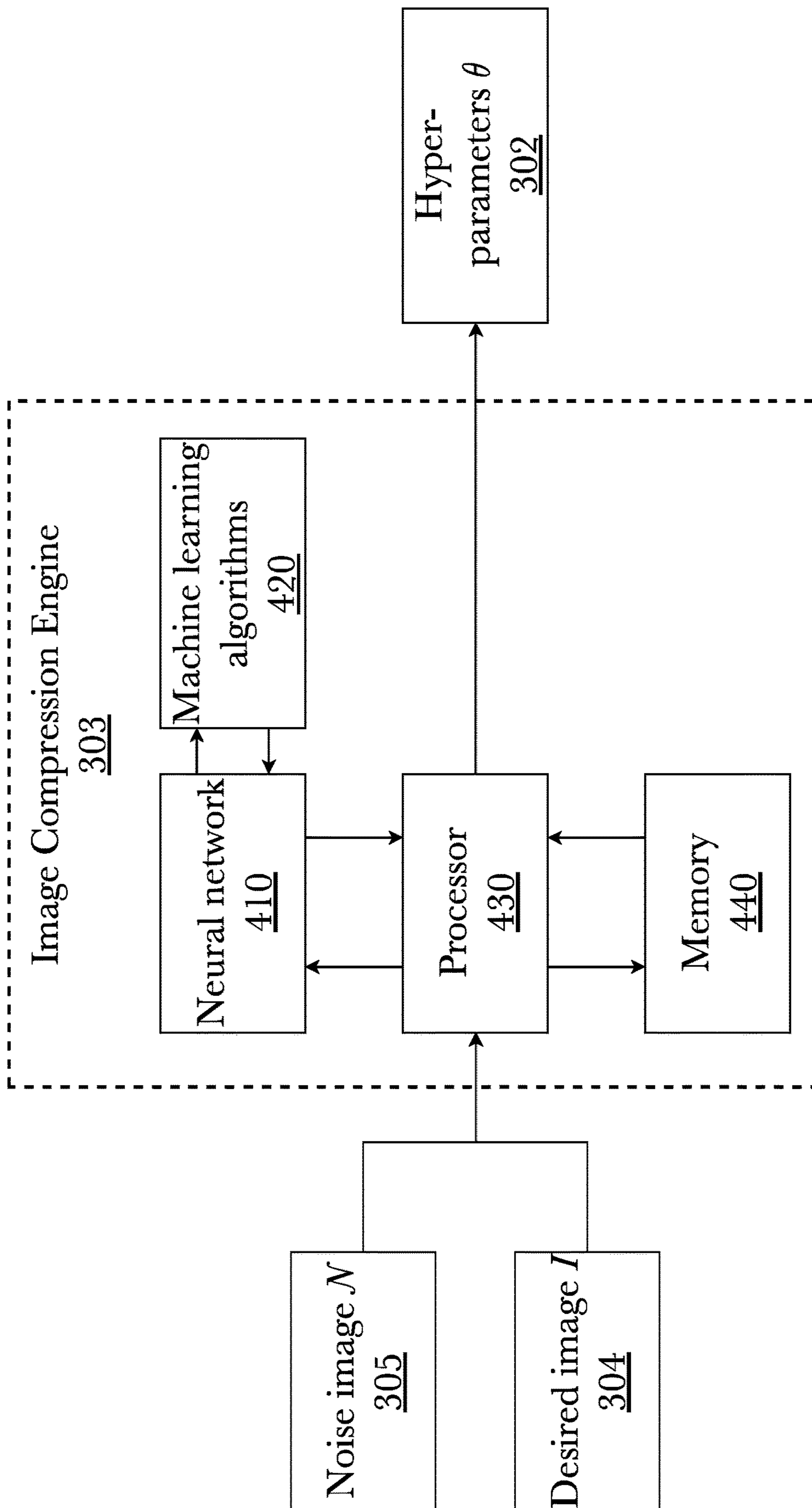


Fig. 4

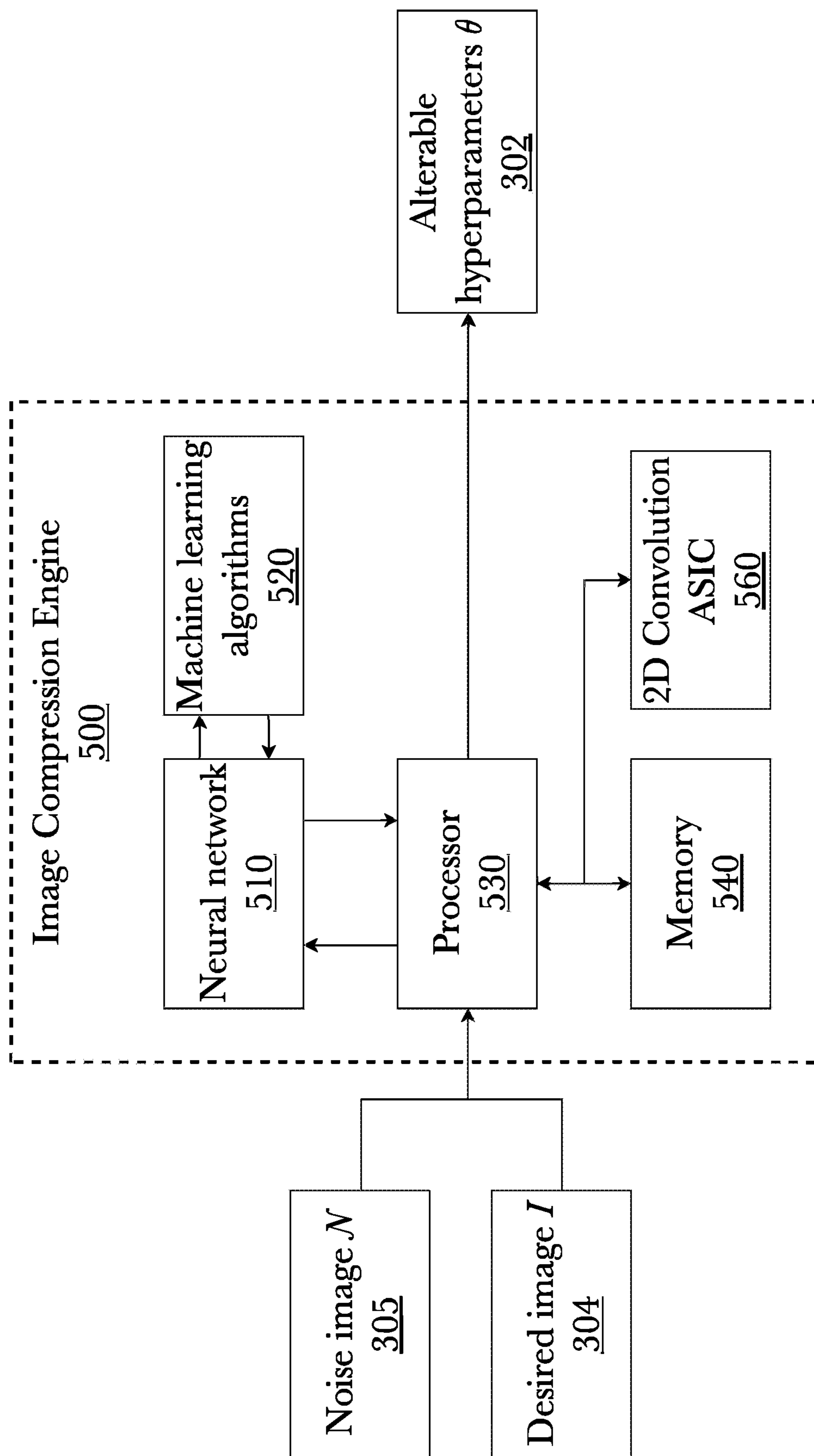
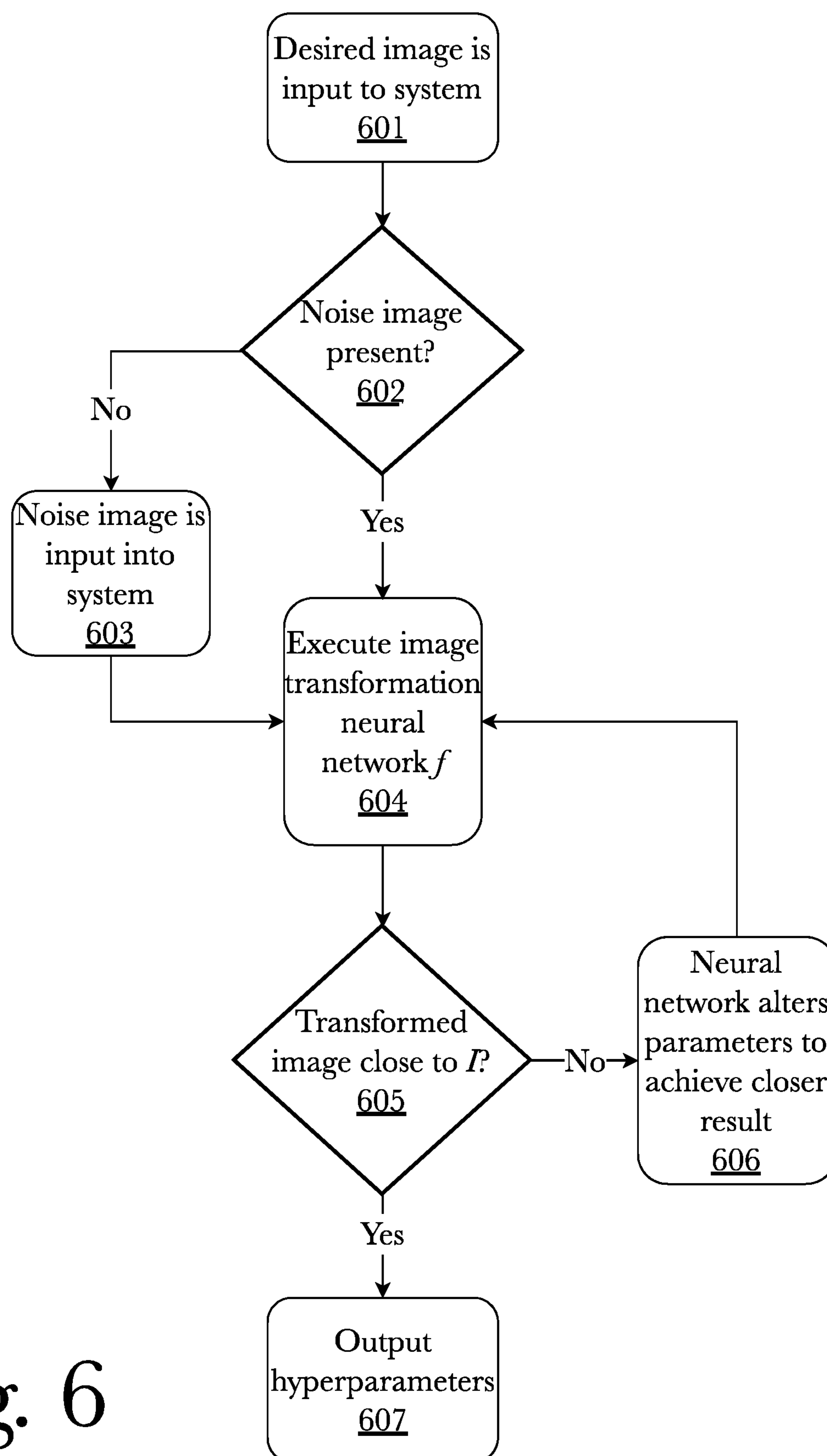


Fig. 5

**Fig. 6**

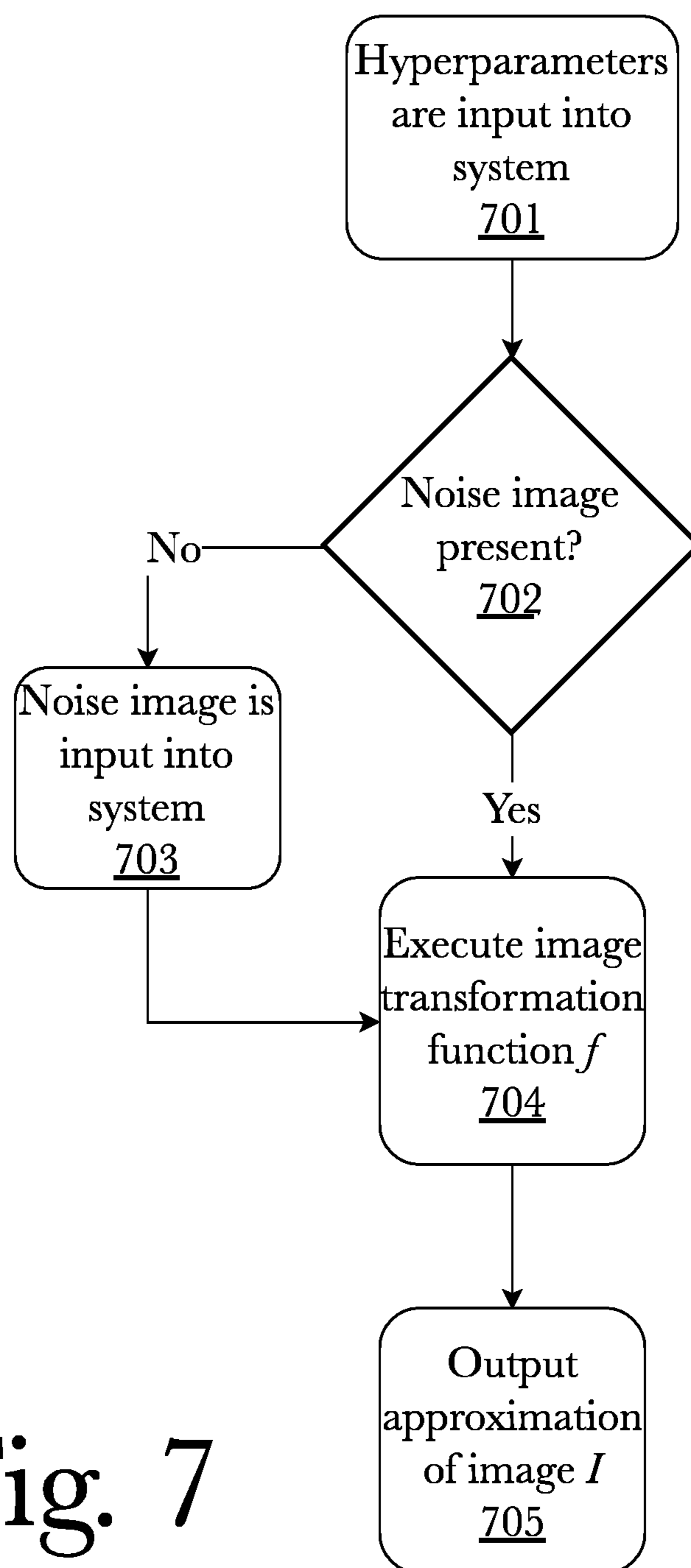
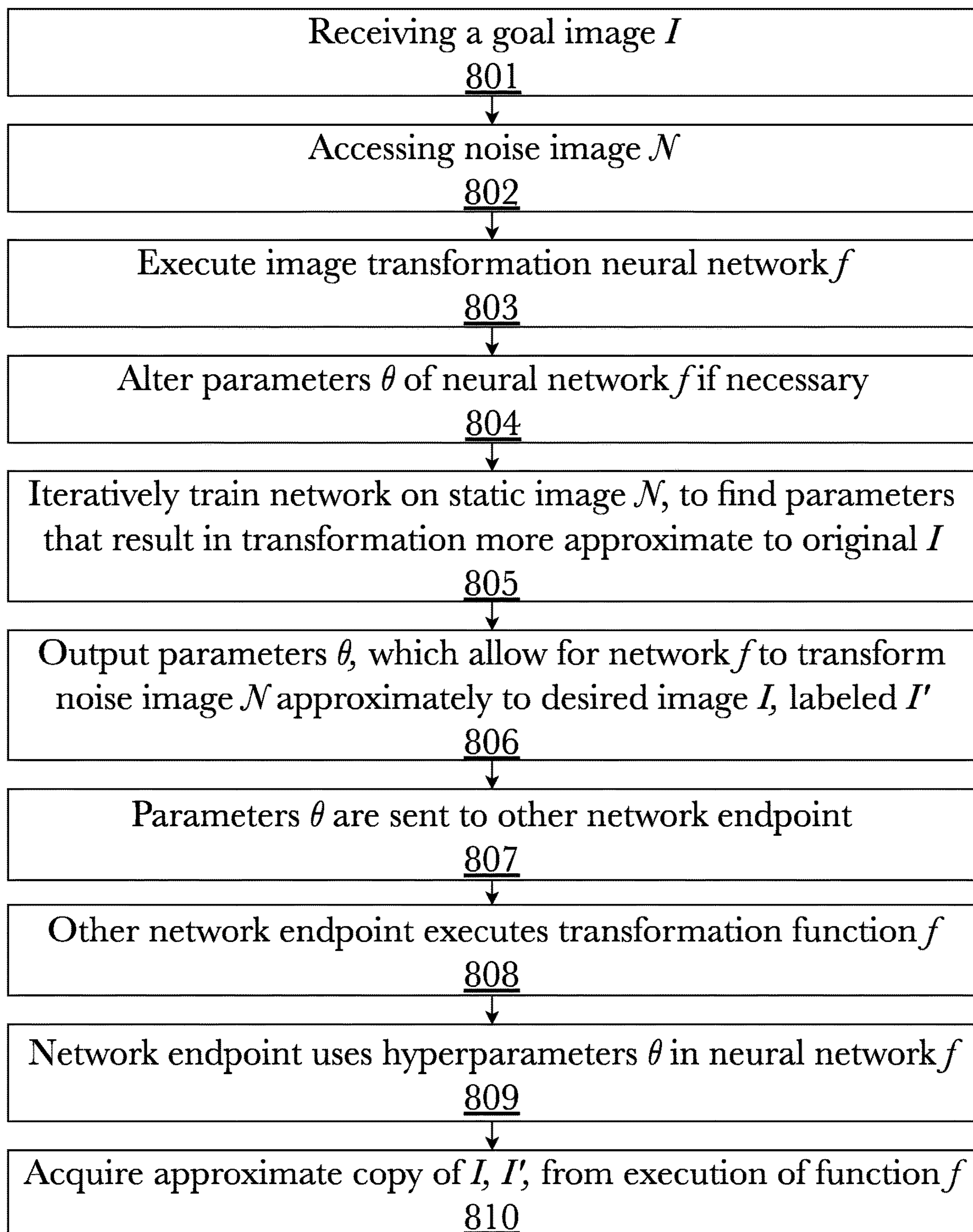


Fig. 7

**Fig. 8**

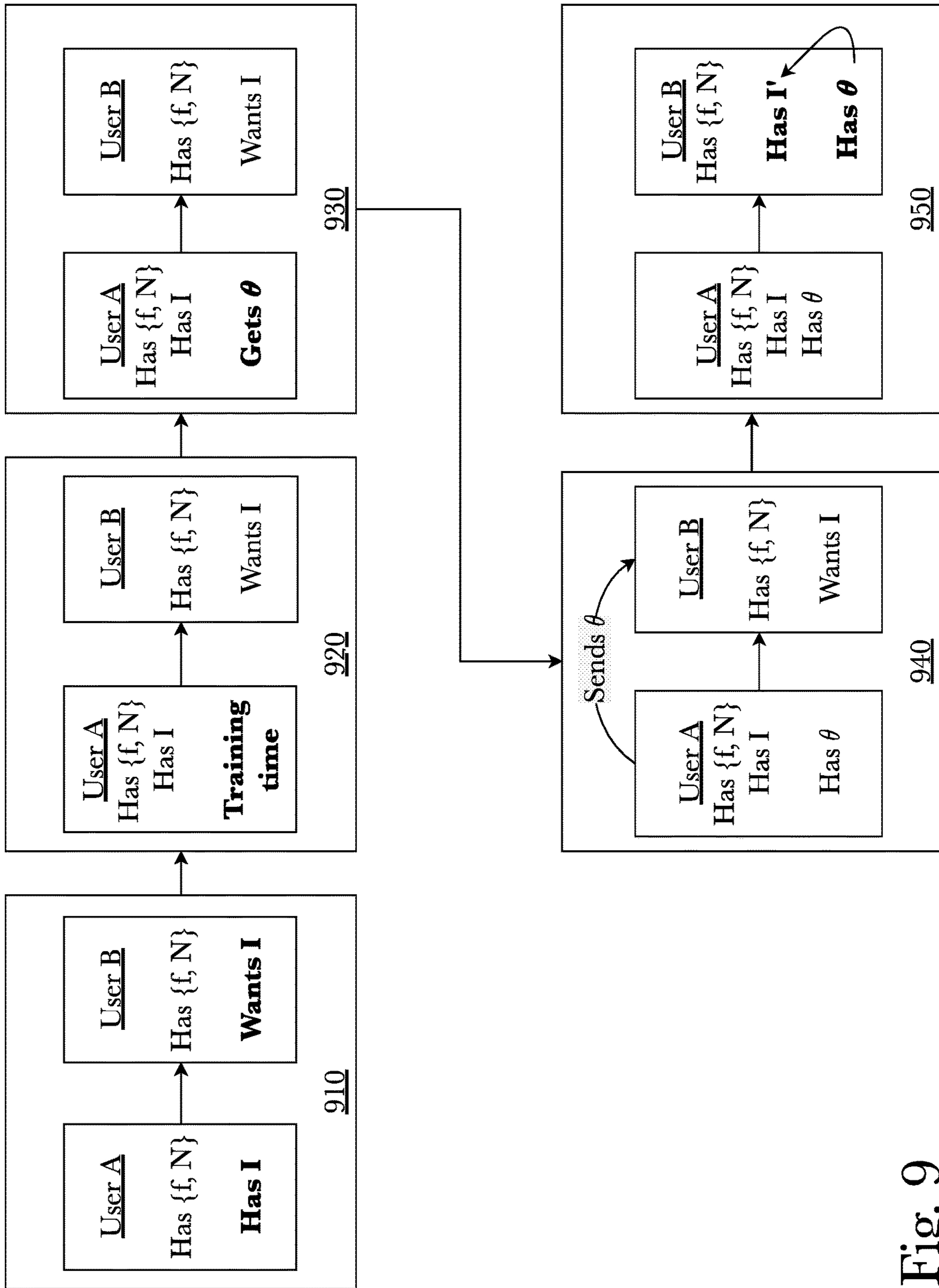
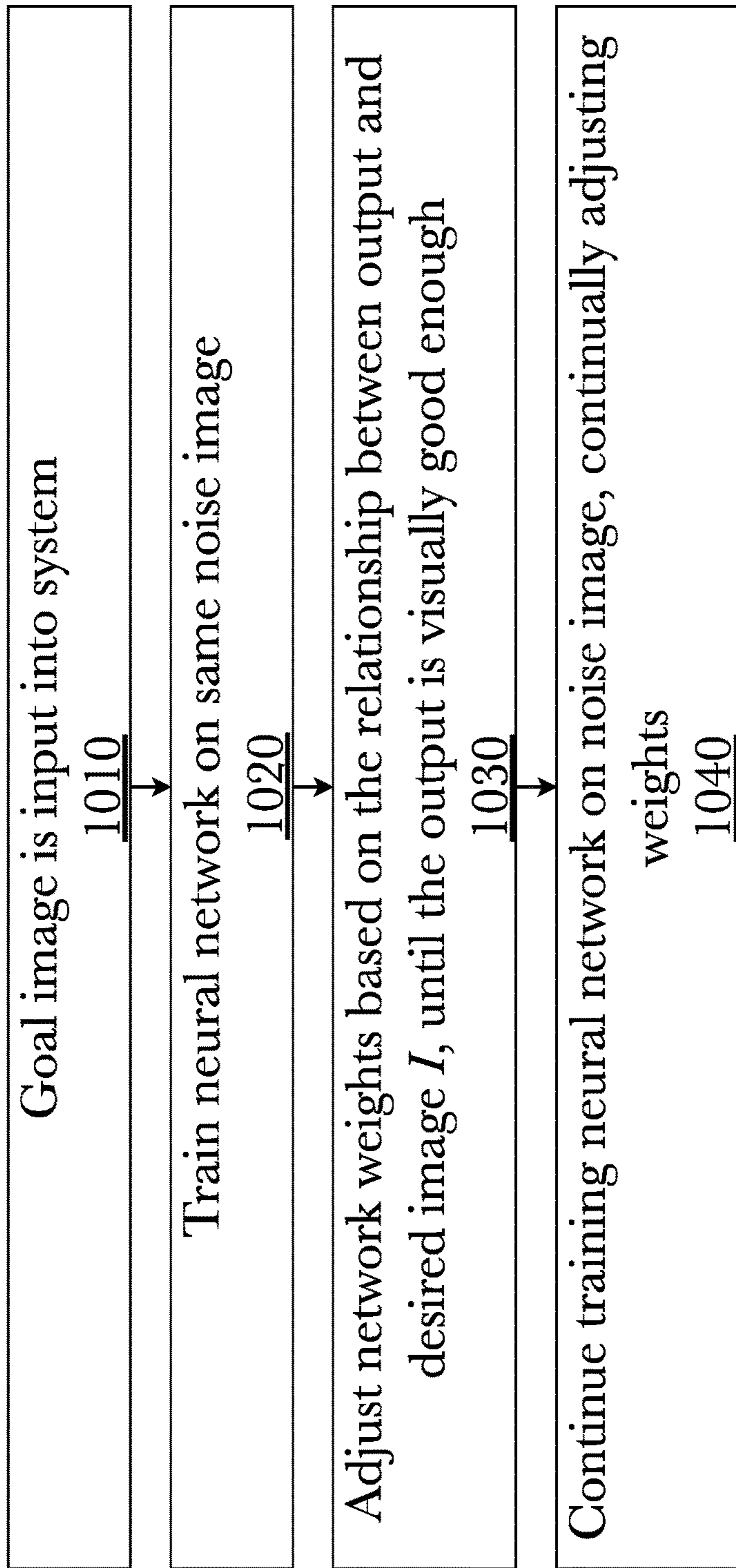


Fig. 9



1050

	Input	Weights	Dataset
Traditional Neural Network:	Flexible	Fixed	Huge
Proposed Neural Network:	Fixed	Flexible	One image

Fig. 10

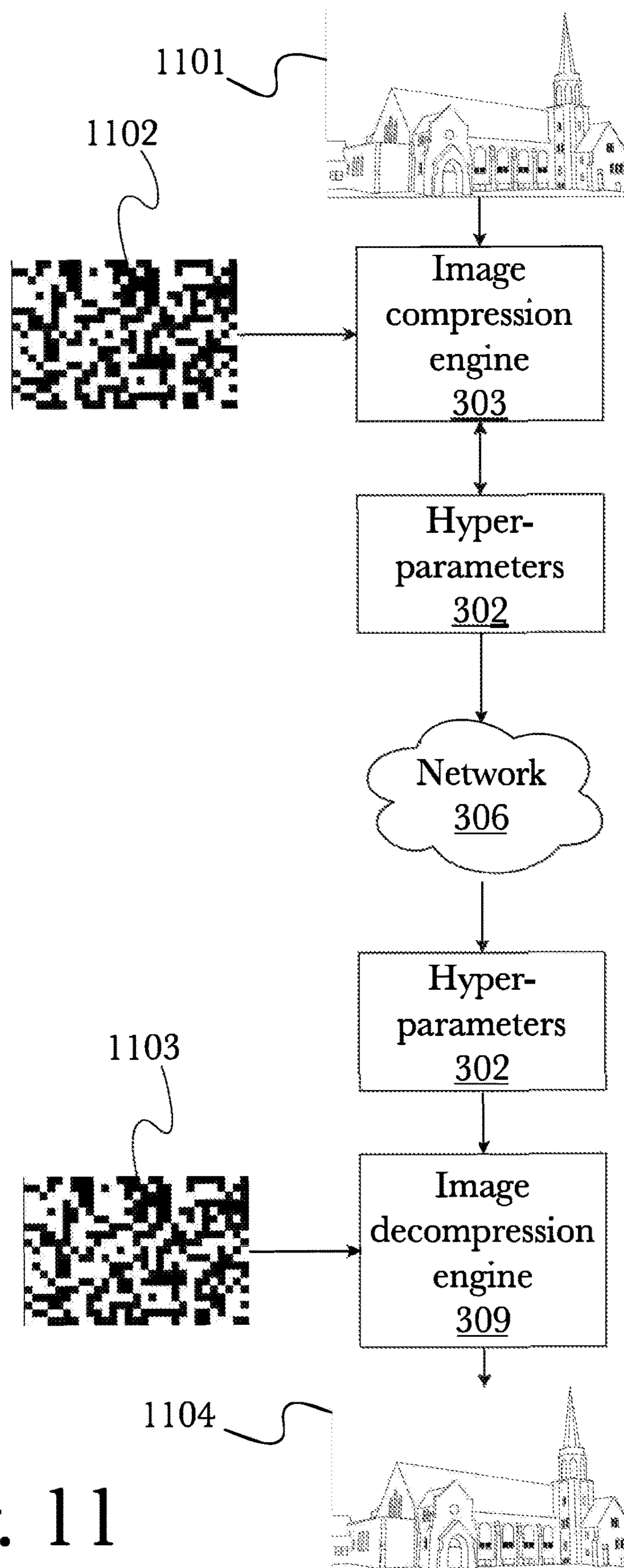


Fig. 11

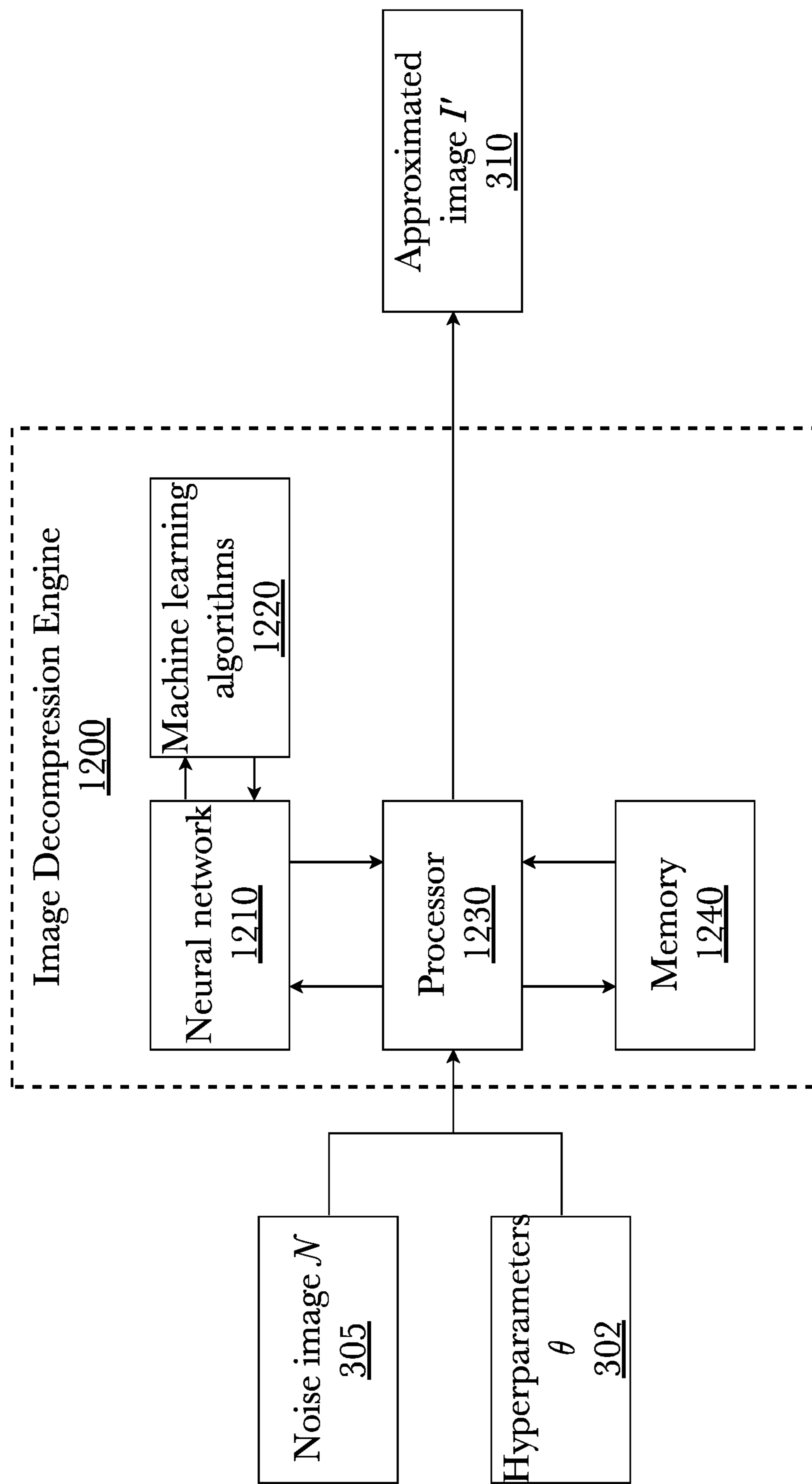


Fig. 12

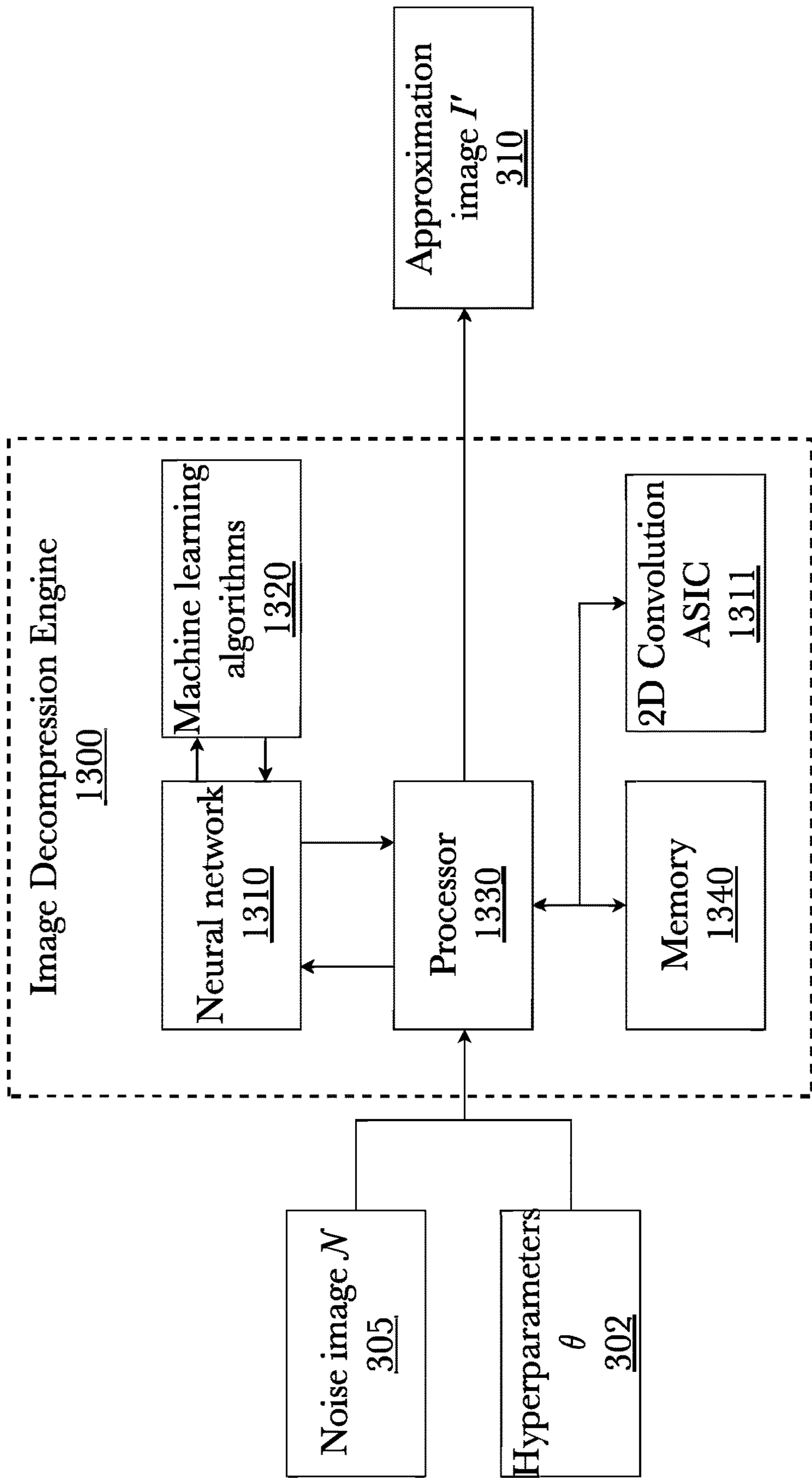


Fig. 13

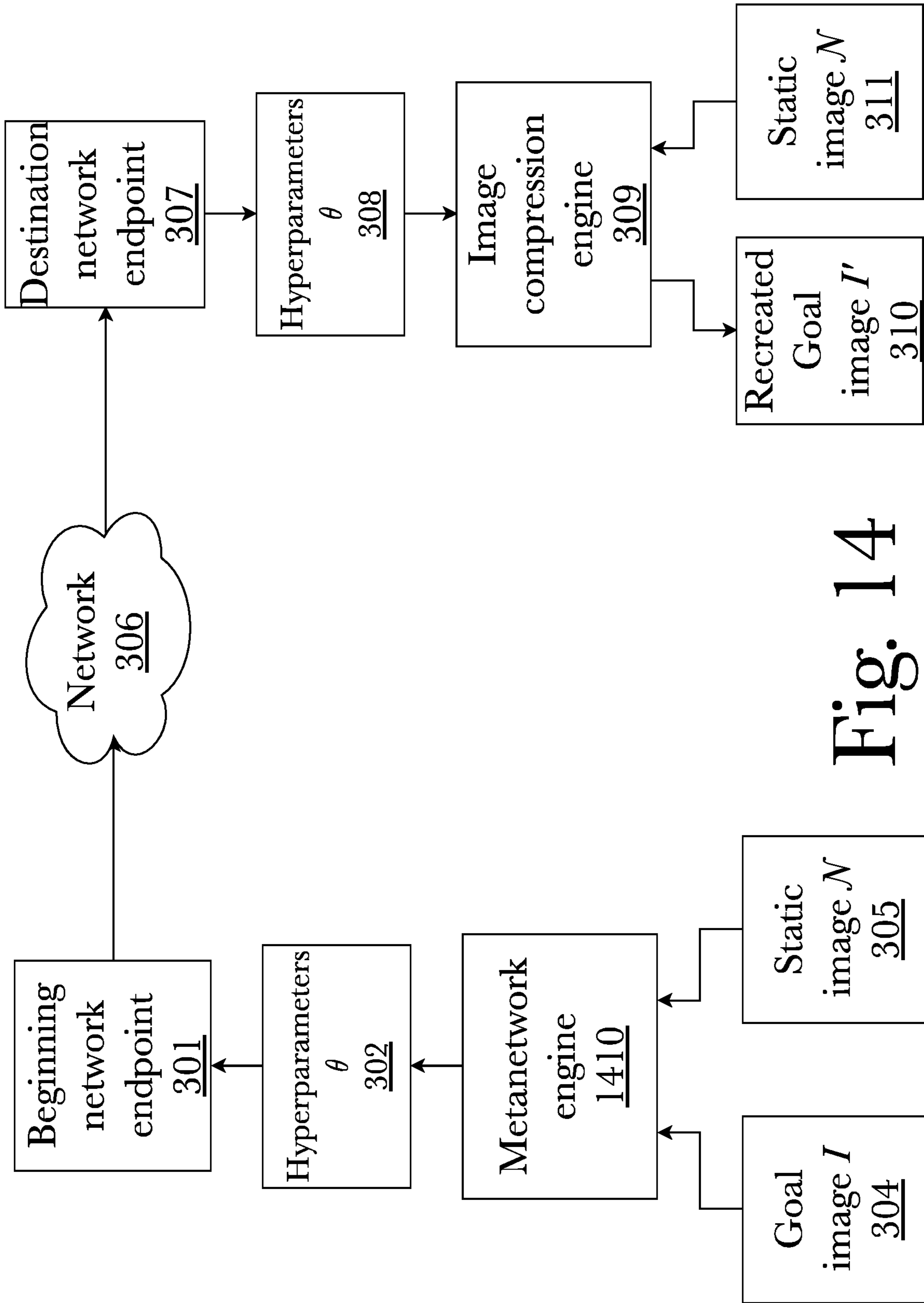


Fig. 14

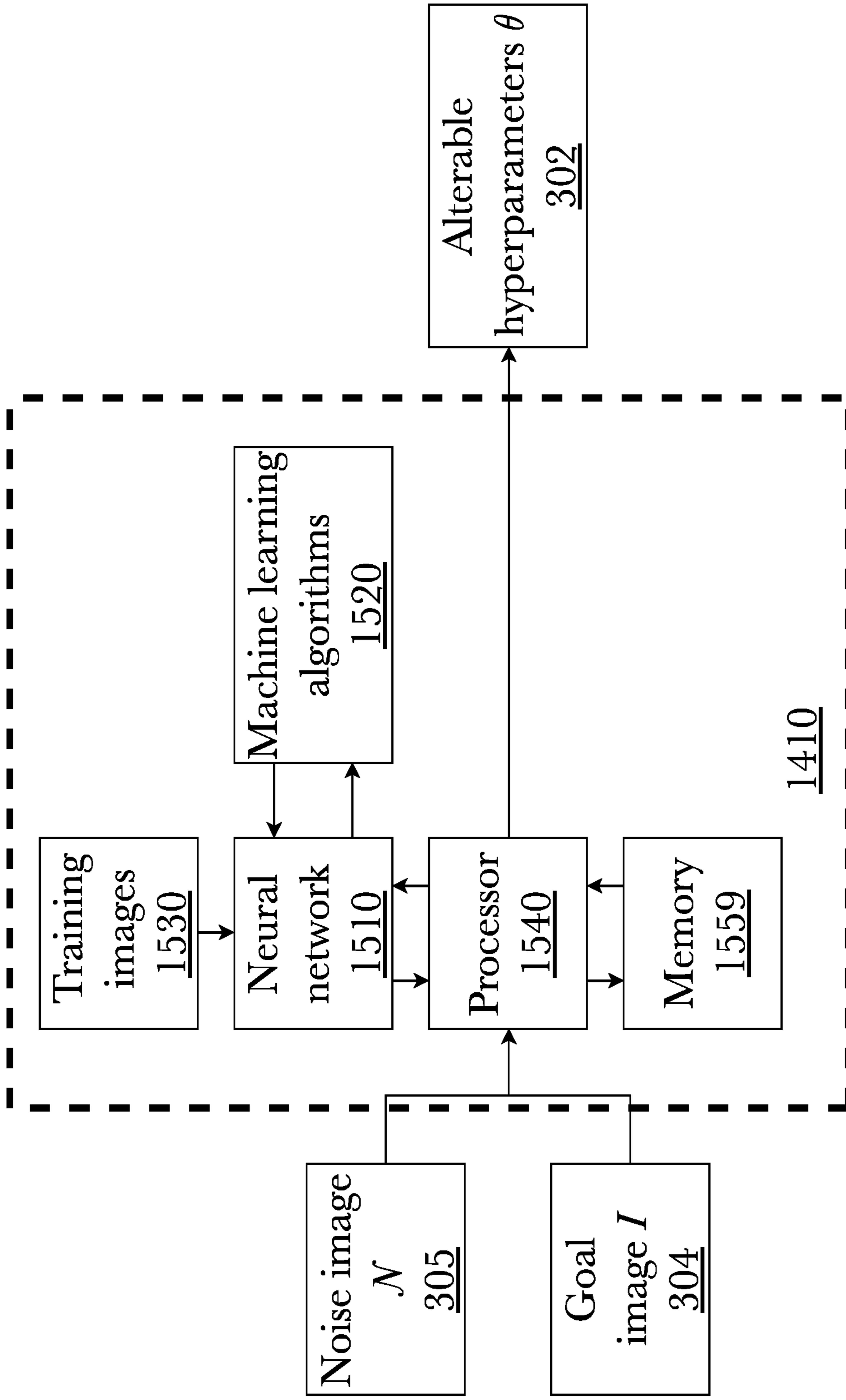


Fig. 15

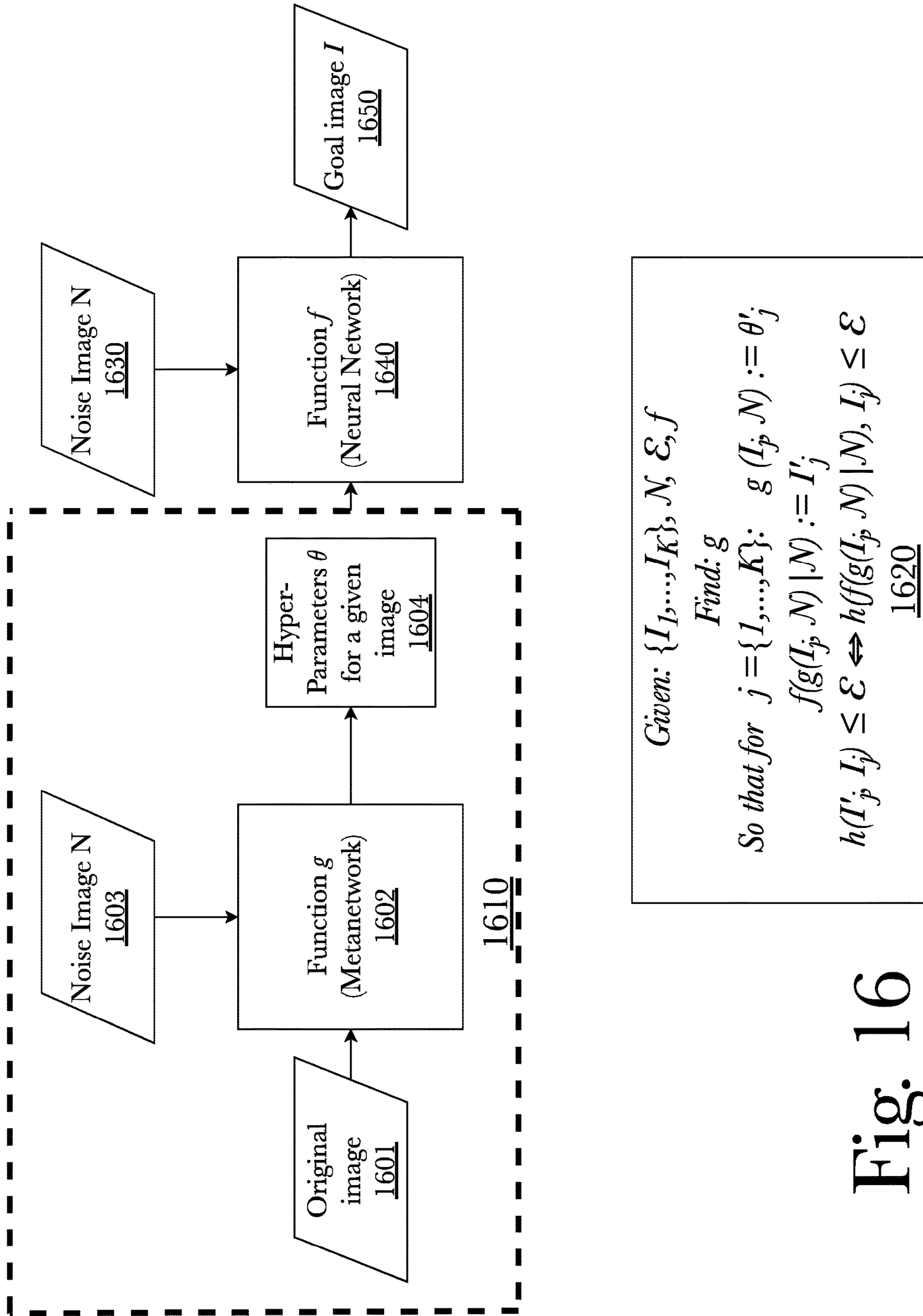


Fig. 16

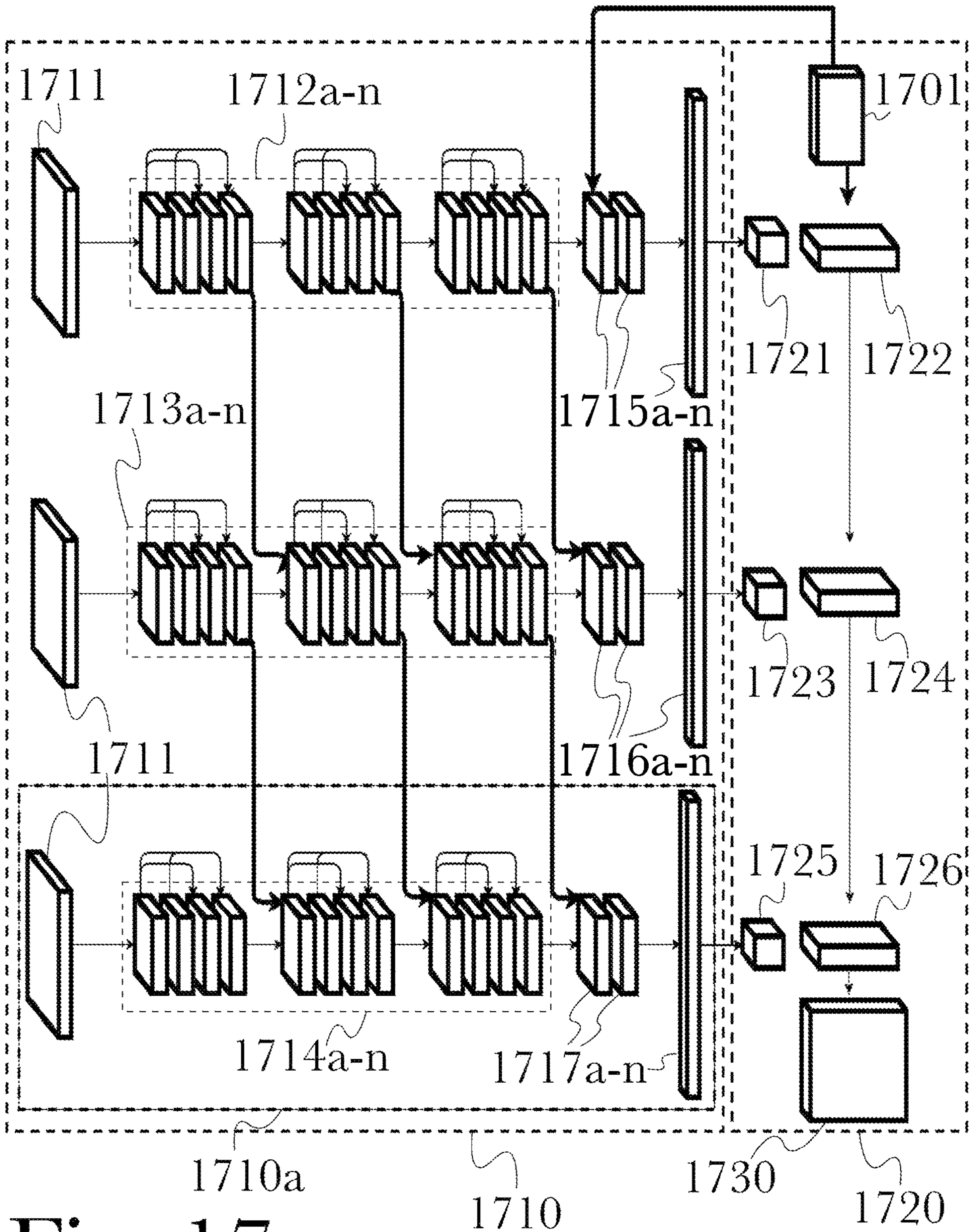


Fig. 17

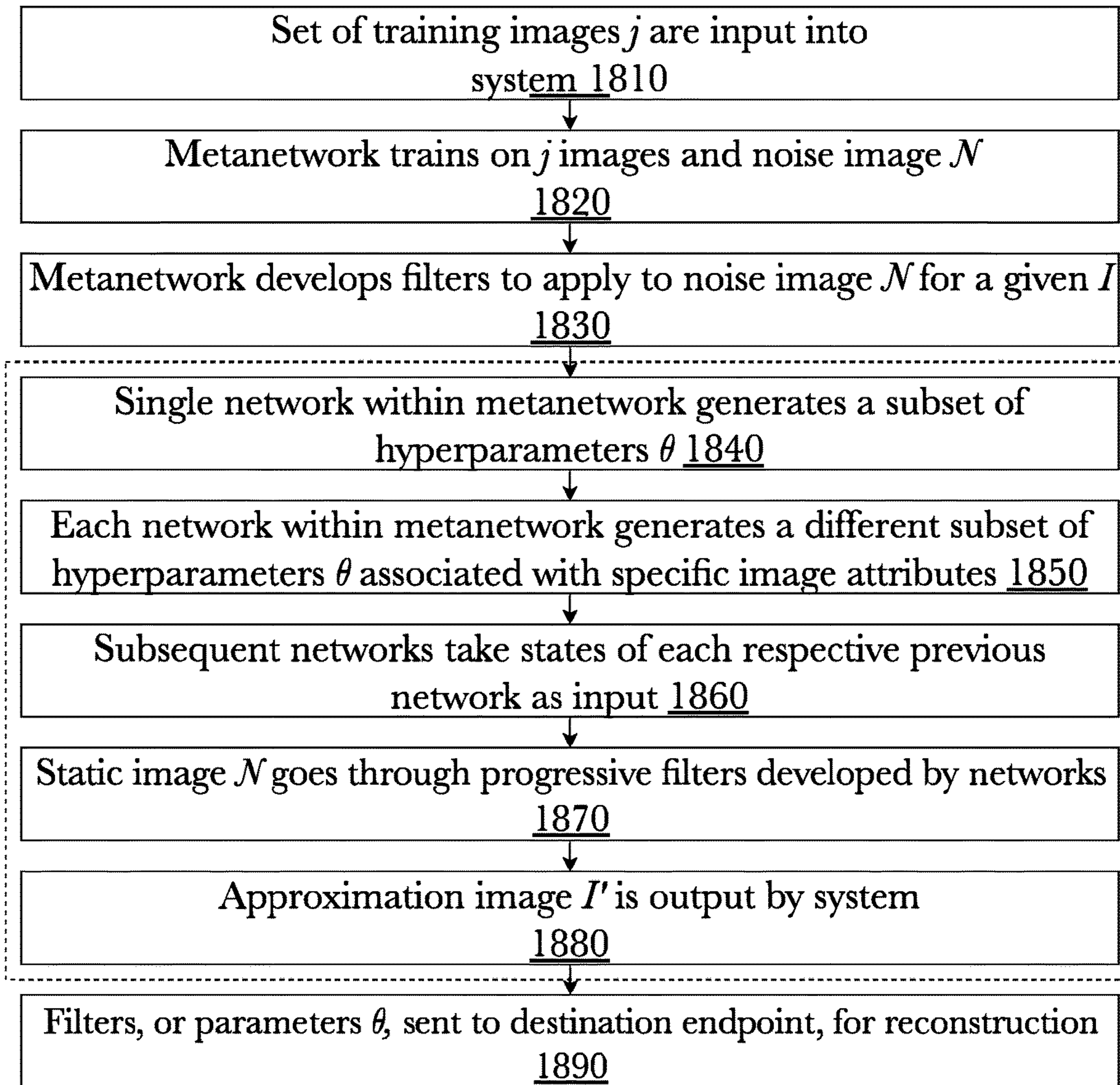


Fig. 18

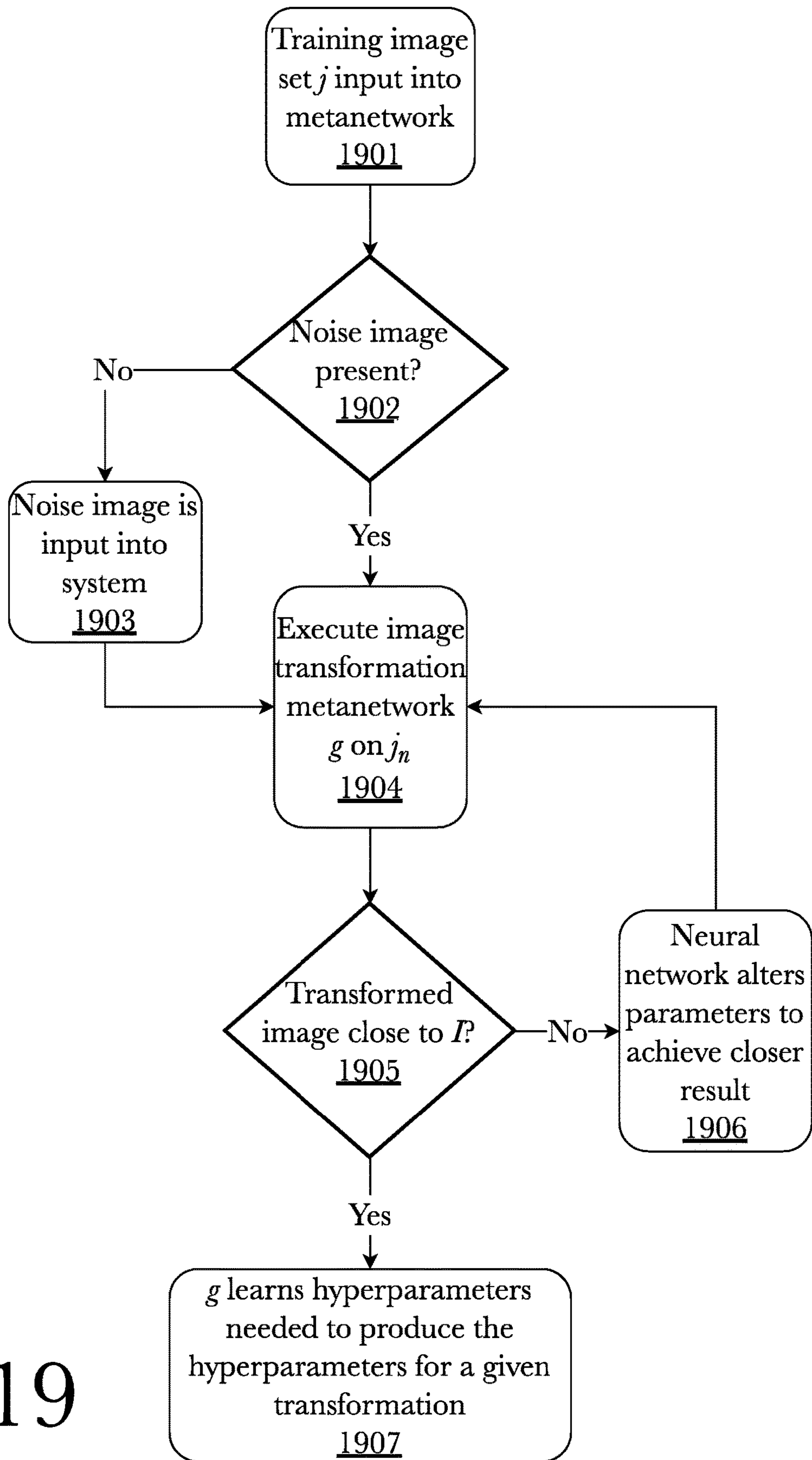


Fig. 19

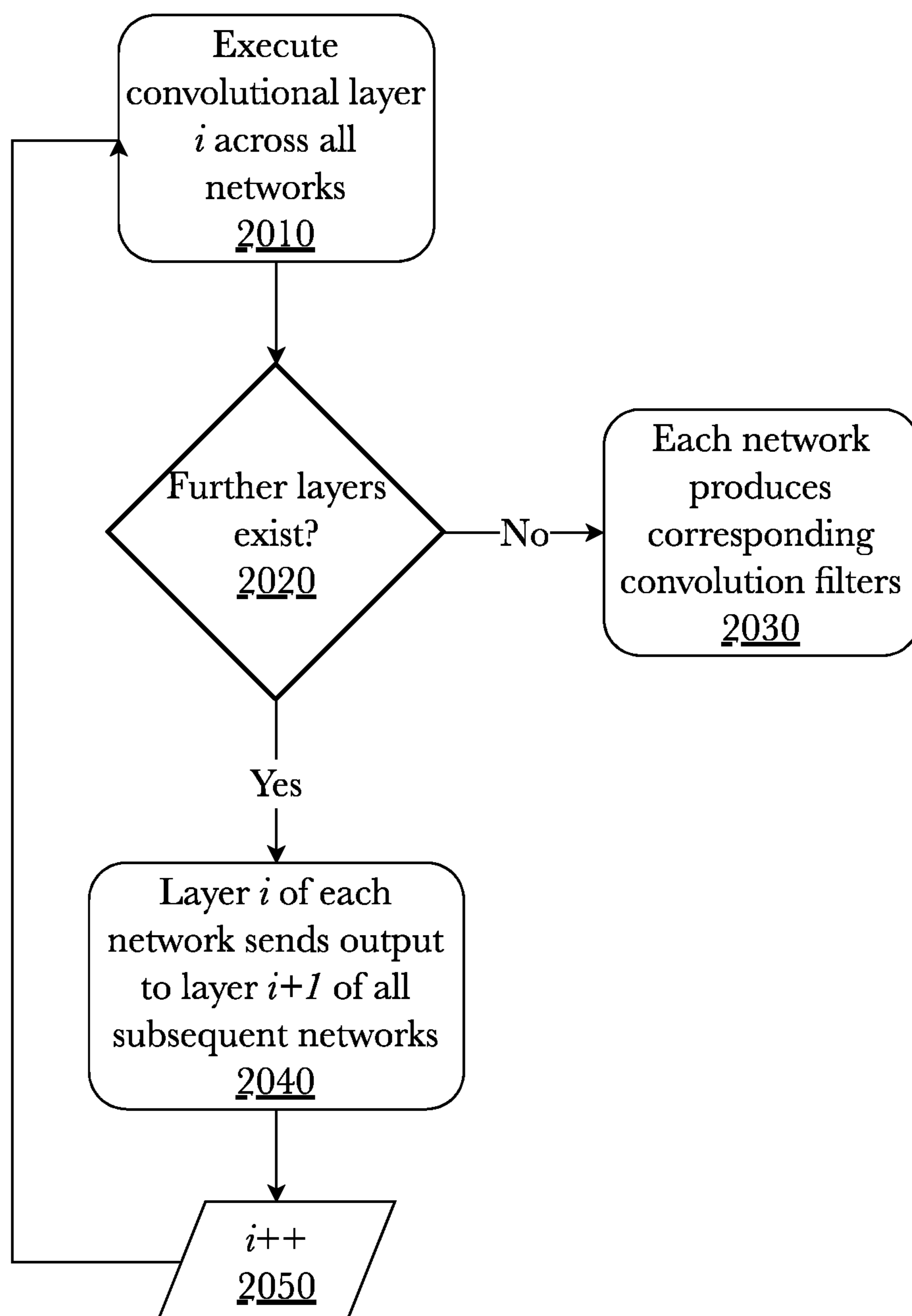


Fig. 20

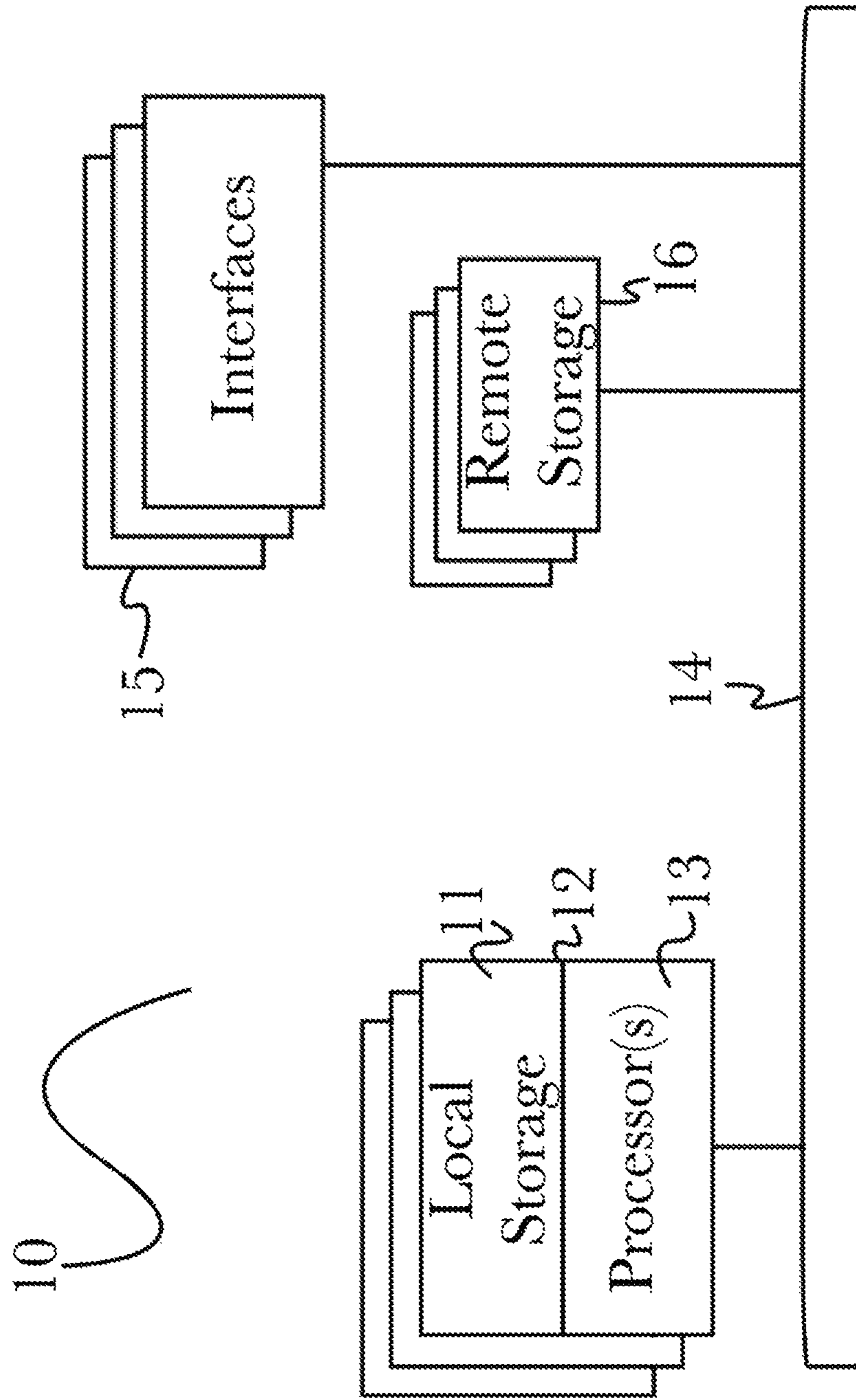


Fig. 21

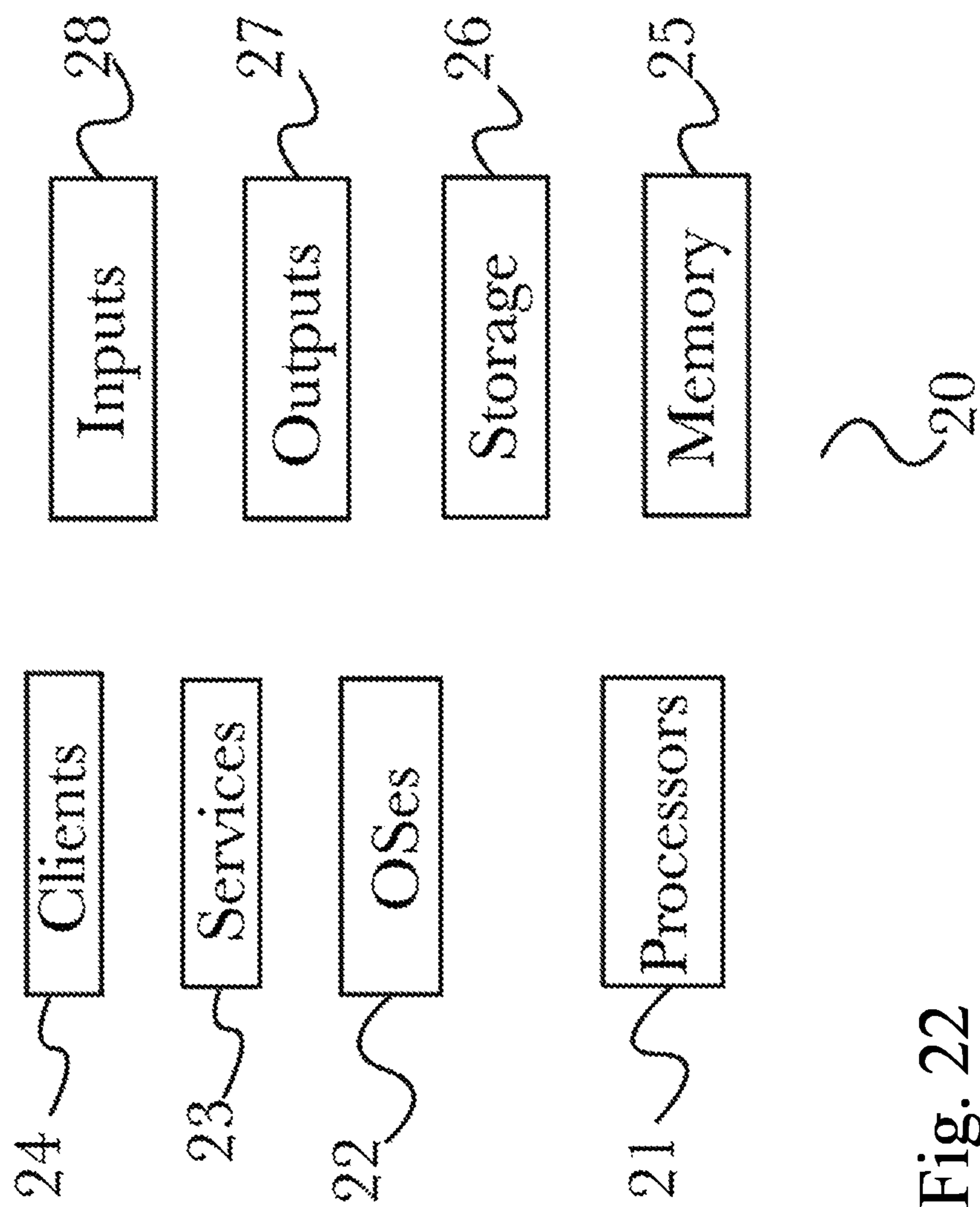


Fig. 22

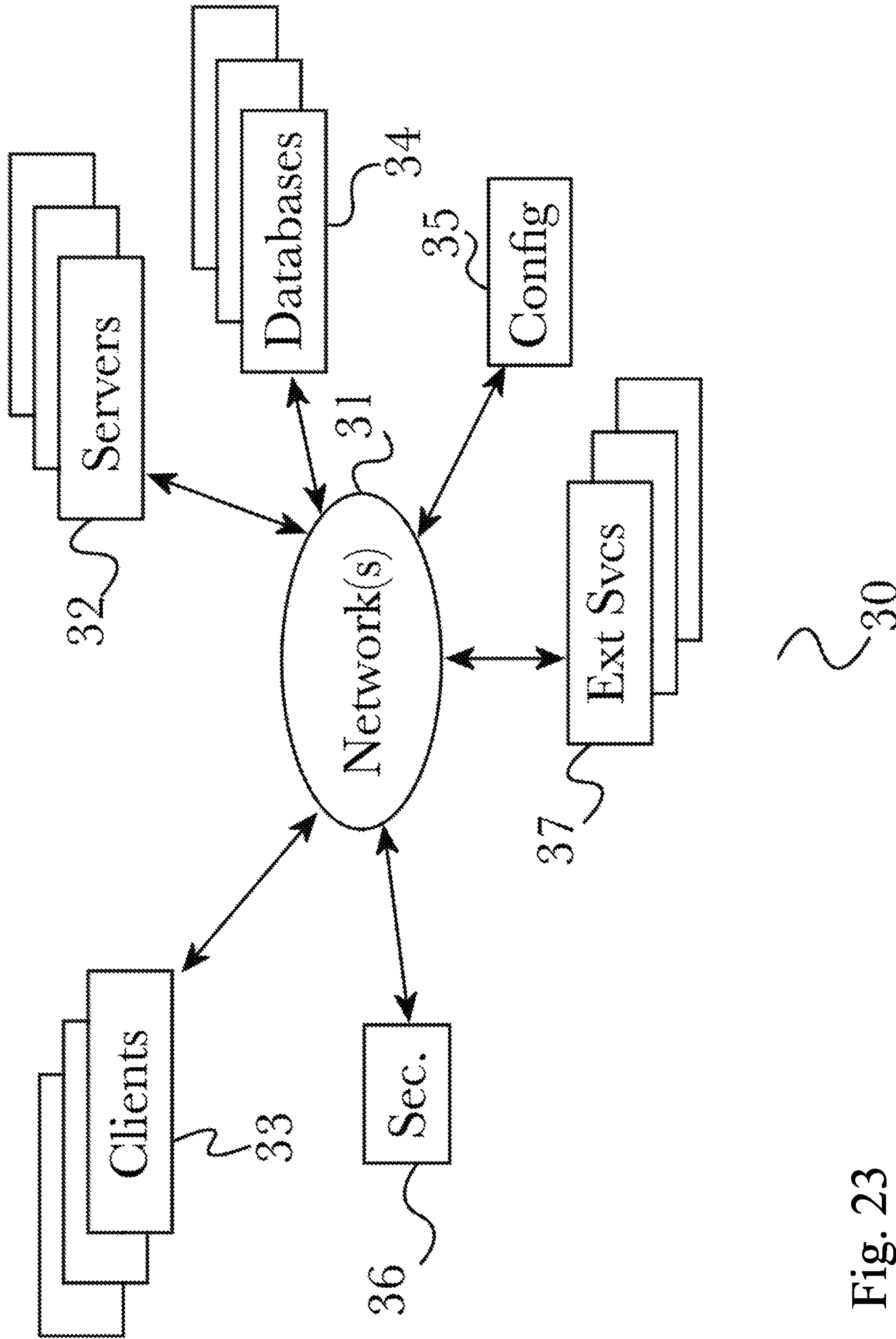


Fig. 23

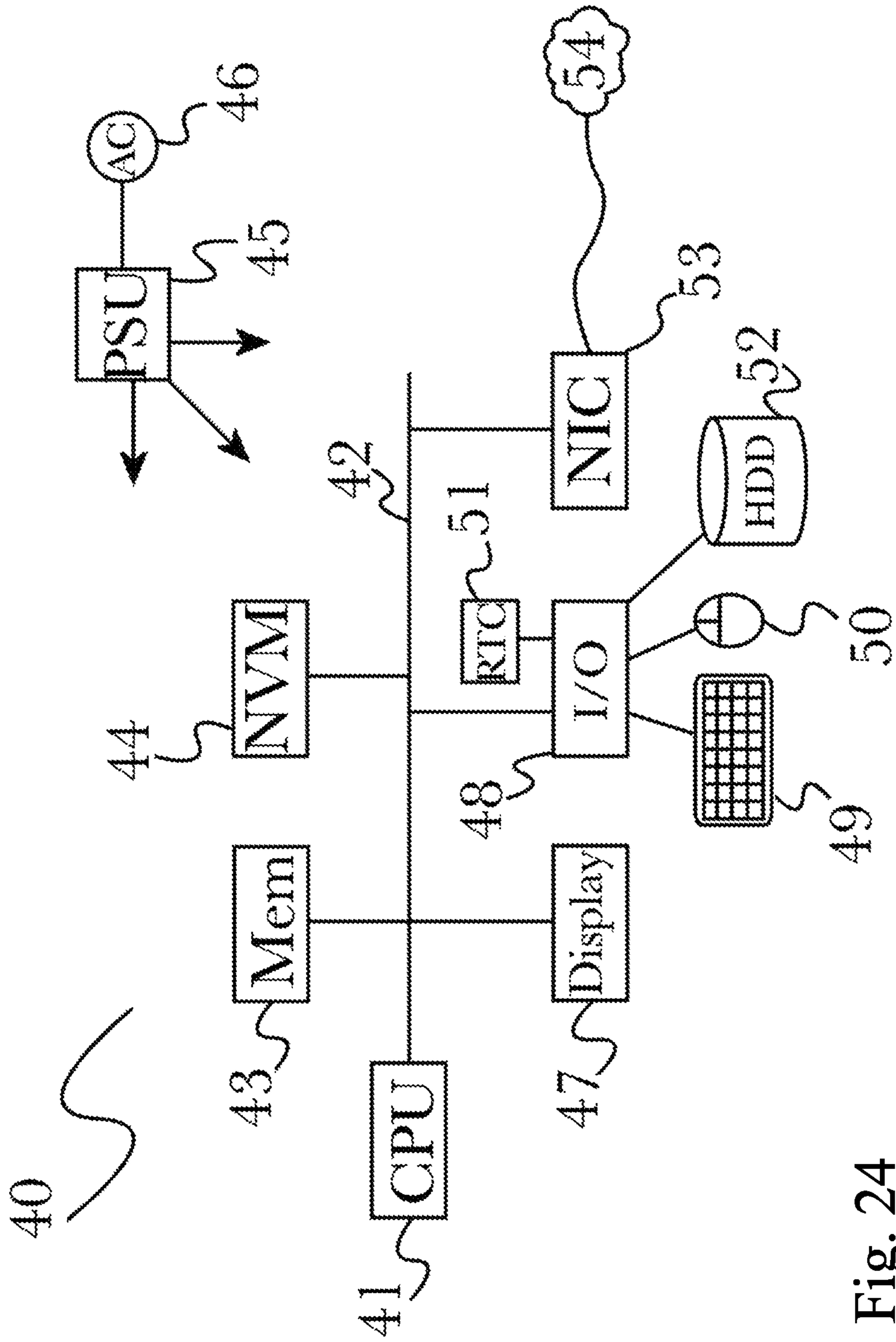


Fig. 24

1

SYSTEM AND METHOD FOR LOSSY IMAGE AND VIDEO COMPRESSION UTILIZING A METANETWORK

CROSS-REFERENCE TO RELATED APPLICATIONS

CROSS-REFERENCE TO RELATED APPLICATIONS		
Application Ser. No.	Date Filed	Title
Current application	Herewith	A SYSTEM AND METHOD FOR LOSSY IMAGE AND VIDEO COMPRESSION UTILIZING A METANETWORK
		Is a continuation-in-part of
16/397,725	Apr. 29, 2019	A SYSTEM AND METHOD FOR LOSSY IMAGE AND VIDEO COMPRESSION AND TRANSMISSION UTILIZING NEURAL NETWORKS

the entire specification of each of which is incorporated herein by reference.

BACKGROUND OF THE INVENTION

Field of the Art

The disclosure relates to the field of data compression, more specifically to the field of lossy image compression utilizing non-generic neural networks.

Discussion of the State of the Art

Since the start of widespread use of the internet a bit more than two decades ago, there has been an exponential growth in the amount of data being transmitted worldwide. In the past decade, even as the amount of data transmitted continues to grow exponentially, video content has accounted for an ever-increasing portion of those data. According to at least one study, video content accounted for 64% of the world's internet traffic in 2014, and is on track to account for up to 85% of the world's internet traffic in 2019.

Given that the amount of data being transmitted continues to grow exponentially, and that most of the world's internet traffic is now video content, video compression technology has become critically important. However, existing video compression technology has not kept pace with the change. The current leading video compression technologies, High Efficiency Video Coding (HVEC, also known as H.265) and its open-source competitor, AV1 are incremental improvements of video compression technologies developed in the early part of this century (i.e., H.264/MPEG-4 AVC and Google's open-source VP coding format). Although the newer video compression standards have improved throughput at the same level of quality versus their older counterparts, that method of video compression may already have hit a point of diminishing returns, and is unlikely to yield significant further improvements.

A method for utilizing a non-generic neural network provides for a significantly reduced amount of data to be transferred for image compression or video frame compression, however such a method, described herein, takes a large amount of time to optimize for finding an optimal network configuration to compress a single image. Therefore, a meta-network which is capable of operating and training on a large data-set, possibly on a powerful platform such as connected computational devices in a datacenter to further

2

improve efficiency, may be able to be trained to quickly find the structure of a neural network which may re-construct an approximation of an image, thereby allowing many corporations or agencies that require the transfer of large amounts of video or images over a network to accomplish this while utilizing a fraction of the bandwidth currently required even with current compression methods.

What is needed is a system and method for image and video compression whose performance dramatically exceeds the performance of current state of the art image and video compression, without taking a large amount of time to compress the image or video.

SUMMARY OF THE INVENTION

Accordingly, the inventor has conceived and reduced to practice, a system and methods for lossy image and video compression that utilizes a metanetwork to generate a set of hyperparameters necessary for an image encoding network to reconstruct the desired image from a given noise image.

According to a preferred embodiment, a system for lossy image and video compression utilizing a metanetwork, comprising: a metanetwork engine comprising a processor, a memory, and a first plurality of programming instructions stored in the memory, wherein the first plurality of programming instructions, when operating on the processor, cause the processor to: receive a desired image; receive a noise image; receive a set of training images; using the set of training images, train a plurality of neural networks to reconstruct each of the set of training images by mapping the noise image to each of the set of training images; store the parameters for each of the plurality of neural networks as a set of metanetwork hyperparameters; use the set of metanetwork hyperparameters as operating parameters for each of the plurality of neural networks; use the plurality of neural networks to map the noise image to the desired image, producing a second set of hyperparameters corresponding to the specific filters produced from the operation of each of the plurality of neural networks, such that the second set of hyperparameters, when applied to the noise image using the neural network, produce an approximation of the desired image within an error that is less than a pre-determined threshold; and store the second set of hyperparameters for use in future image mapping operations.

According to another preferred embodiment, a method for lossy image compression utilizing a metanetwork, comprising the steps of: receiving a desired image; receiving a noise image; receiving a set of training images; using the set of training images to train a plurality of neural networks to reconstruct each of the set of training images by mapping the noise image to each of the set of training images; storing the parameters for each of the plurality of neural networks as a set of metanetwork hyperparameters; using the set of metanetwork hyperparameters as operating parameters for each of the plurality of neural networks; using the plurality of neural networks to map the noise image to the desired image, producing a second set of hyperparameters corresponding to the specific filters produced from the operation of each of the plurality of neural networks, such that the second set of hyperparameters, when applied to the noise image using the neural network, produce an approximation of the desired image within an error that is less than a pre-determined threshold; and storing the second set of hyperparameters for use in future image mapping operations.

BRIEF DESCRIPTION OF THE DRAWING
FIGURES

The accompanying drawings illustrate several aspects and, together with the description, serve to explain the principles of the invention according to the aspects. It will be appreciated by one skilled in the art that the particular arrangements illustrated in the drawings are merely exemplary, and are not to be considered as limiting of the scope of the invention or the claims herein in any way.

FIG. 1 (PRIOR ART) is a method diagram illustrating the HVEC video compression and the similar BPG image compression methodology.

FIG. 2 is a diagram illustrating a flow of objects from compression to decompression stages, illustrating the function used for the disclosed invention, according to an embodiment.

FIG. 3 is a system diagram of high-level components used in the operation of a system for lossy image compression utilizing non-generic neural networks, according to a preferred embodiment.

FIG. 4 is a system diagram of an image compression engine, according to an aspect.

FIG. 5 is a system diagram of an image compression engine as it is utilized for image compression, utilizing a 2D convolution Application Specific Integrated Circuit ("ASIC"), according to a preferred aspect.

FIG. 6 is a flowchart illustrating the processing and compression of data through the system, according to an aspect.

FIG. 7 is a flowchart illustrating the processing and decompression of data through the system, according to an aspect.

FIG. 8 is a method diagram of high-level components used in the operation of a system for lossy image compression utilizing non-generic neural networks, according to a preferred embodiment.

FIG. 9 is a state diagram of two users, one encoding and one decoding an image, using the disclosed compression system, according to a preferred aspect.

FIG. 10 is a method diagram illustrating training a neural network with a single image, as opposed to a plurality of images to learn from, resulting in a non-general approach to image compression, according to a preferred embodiment.

FIG. 11 is a diagram illustrating a line-drawn image, and a static image, going into an image compression engine, parameters being identified which allow for a static image to be approximately translated to an initial non-static image, and relayed to a second system over a network to re-generate an approximation of the original image from a static image.

FIG. 12 is a system diagram of an image compression engine used for decompressing an image, according to a preferred aspect.

FIG. 13 is a system diagram of an image compression engine utilizing a specialized 2D convolution Application Specific Integrated Circuit ("ASIC").

FIG. 14 is a system diagram of high-level components used in the operation of a system using a metanetwork to achieve image or video compression, and decompression, according to a preferred embodiment.

FIG. 15 is a system diagram of a metanetwork engine, used to train a metanetwork with a set of training images, and used to compress and find filters that may be used to transform a noise image into an approximation of an input image, according to a preferred embodiment.

FIG. 16 is a dataflow diagram of a system for lossy compression utilizing a metanetwork to train, compress, and

send data for decompression to a system utilizing a specific neural network configuration, according to an embodiment.

FIG. 17 is a system diagram of multiple individual networks communicating with each other within a metanetwork, to produce a sequence of convolutional filters to be applied to a noise image, to progressively transform the image into an approximation of an input image, according to an embodiment.

FIG. 18 is a method diagram illustrating steps required for lossy compression of images and video using a metanetwork.

FIG. 19 is a flowchart of the steps taken for a single network within a metanetwork to train on a set of images and produce a network operating as a portion of function g, for neural network hyperparameter prediction of an image encoding network f, according to an embodiment.

FIG. 20 is a flowchart of the process of multiple networks communicating within a metanetwork for the purposes of cross-training and developing progressive filters to transform a static image with, to help alleviate the vanishing gradient problem, according to an embodiment.

FIG. 21 is a block diagram illustrating an exemplary hardware architecture of a computing device.

FIG. 22 is a block diagram illustrating an exemplary logical architecture for a client device.

FIG. 23 is a block diagram showing an exemplary architectural arrangement of clients, servers, and external services.

FIG. 24 is another block diagram illustrating an exemplary hardware architecture of a computing device.

DETAILED DESCRIPTION

The inventor has conceived, and reduced to practice, a system and methods for lossy image and video compression that utilizes a metanetwork.

One or more different aspects may be described in the present application. Further, for one or more of the aspects described herein, numerous alternative arrangements may be described; it should be appreciated that these are presented for illustrative purposes only and are not limiting of the aspects contained herein or the claims presented herein in any way. One or more of the arrangements may be widely applicable to numerous aspects, as may be readily apparent from the disclosure. In general, arrangements are described in sufficient detail to enable those skilled in the art to practice one or more of the aspects, and it should be appreciated that other arrangements may be utilized and that structural, logical, software, electrical and other changes may be made without departing from the scope of the particular aspects. Particular features of one or more of the aspects described herein may be described with reference to one or more particular aspects or figures that form a part of the present disclosure, and in which are shown, by way of illustration, specific arrangements of one or more of the aspects. It should be appreciated, however, that such features are not limited to usage in the one or more particular aspects or figures with reference to which they are described. The present disclosure is neither a literal description of all arrangements of one or more of the aspects nor a listing of features of one or more of the aspects that must be present in all arrangements.

Headings of sections provided in this patent application and the title of this patent application are for convenience only, and are not to be taken as limiting the disclosure in any way.

Devices that are in communication with each other need not be in continuous communication with each other, unless expressly specified otherwise. In addition, devices that are in communication with each other may communicate directly or indirectly through one or more communication means or intermediaries, logical or physical.

A description of an aspect with several components in communication with each other does not imply that all such components are required. To the contrary, a variety of optional components may be described to illustrate a wide variety of possible aspects and in order to more fully illustrate one or more aspects. Similarly, although process steps, method steps, algorithms or the like may be described in a sequential order, such processes, methods and algorithms may generally be configured to work in alternate orders, unless specifically stated to the contrary. In other words, any sequence or order of steps that may be described in this patent application does not, in and of itself, indicate a requirement that the steps be performed in that order. The steps of described processes may be performed in any order practical. Further, some steps may be performed simultaneously despite being described or implied as occurring non-simultaneously (e.g., because one step is described after the other step). Moreover, the illustration of a process by its depiction in a drawing does not imply that the illustrated process is exclusive of other variations and modifications thereto, does not imply that the illustrated process or any of its steps are necessary to one or more of the aspects, and does not imply that the illustrated process is preferred. Also, steps are generally described once per aspect, but this does not mean they must occur once, or that they may only occur once each time a process, method, or algorithm is carried out or executed. Some steps may be omitted in some aspects or some occurrences, or some steps may be executed more than once in a given aspect or occurrence.

When a single device or article is described herein, it will be readily apparent that more than one device or article may be used in place of a single device or article. Similarly, where more than one device or article is described herein, it will be readily apparent that a single device or article may be used in place of the more than one device or article.

The functionality or the features of a device may be alternatively embodied by one or more other devices that are not explicitly described as having such functionality or features. Thus, other aspects need not include the device itself.

Techniques and mechanisms described or referenced herein will sometimes be described in singular form for clarity. However, it should be appreciated that particular aspects may include multiple iterations of a technique or multiple instantiations of a mechanism unless noted otherwise. Process descriptions or blocks in figures should be understood as representing modules, segments, or portions of code which include one or more executable instructions for implementing specific logical functions or steps in the process. Alternate implementations are included within the scope of various aspects in which, for example, functions may be executed out of order from that shown or discussed, including substantially concurrently or in reverse order, depending on the functionality involved, as would be understood by those having ordinary skill in the art.

Definitions

“Artificial intelligence” or “AI” as used herein means a computer system or component that has been programmed in such a way that it mimics some aspect or aspects of

cognitive functions that humans associate with human intelligence, such as learning, problem solving, and decision-making. Examples of current AI technologies include understanding human speech, competing successfully in strategic games such as chess and Go, autonomous operation of vehicles, complex simulations, and interpretation of complex data such as images and video.

“Function,” “image transformation function,” “image transformation network,” and “image transformation neural network” as used herein mean the use of a neural network as a function to transform an image or to re-create an approximation of an image. The transformation is based on mapping of a noise image to a target image, and adjustment of the weights of differing variables within the function, which may also be referred to as hyperparameters. Hyperparameters may also be used as inputs to the function, along with the noise image, to re-create an approximation of the image.

“Hyperparameter” as used herein means the parameters of a function that maps a noise image to a target image within a specified margin of error. When an image is input into the function for mapping at a source location, hyperparameters will be the output. The hyperparameters may then be transferred to a destination location, wherein the hyperparameters will be the input to the same (or similar) function with the same noise image, and the output of the function will be an approximation of the desired image at the destination location.

“Goal image”, also referred to interchangeably as a “target image” or “desired image” as used herein means a digital representation of an image. This digital representation may be in any image file format, of which there already exist a great number. Image files are commonly classified as either raster-based formats (i.e., formats in which the content of pixels or areas of the image are specified) or vector-based (i.e., formats in which shapes and their relationships are specified without regard to specific pixels, areas, or image sizes). For exemplary purposes, a non-exhaustive and non-limiting list of raster-based formats includes Microsoft Windows bitmap (bmp), CompuServe Graphics Interchange Format (gif), Joint Photographic Experts Group JFIF format (jpg or jpeg), Portable Graphics Network (png), and Tagged Image File Format (tif or tiff). For exemplary purposes, a non-exhaustive and non-limiting list of vector-based formats includes Adobe Illustrator File (ai), CorelDRAW Image File (cdr), Scalable Vector Graphics File (svg), and Microsoft Visio Drawing File (vsd).

“Image transformation” as used herein means the act of transforming any image, or even a blank or “empty” image file, into any other image. This may be done with numerous different techniques, in any combination, including specific pixel alterations, the use of vector graphics based on mathematical equations (for example, graphing curves with equations which yield perfect resolution no matter the zoom on the image itself, since it is specified with an equation rather than represented by specific pixels at a given resolution), or examining and altering groups of pixels such as for border detection and alteration. Altering the positions of pixels of data in an image, but keeping the data otherwise unchanged, for example rotating an image, is another example of an image transformation.

“Machine learning” as used herein is an aspect of artificial intelligence in which the computer system or component can modify its behavior or understanding without being explicitly programmed to do so. Machine learning algorithms develop models of behavior or understanding based on information fed to them as training sets, and can modify those models based on new incoming information. An

example of a machine learning algorithm is AlphaGo, the first computer program to defeat a human world champion in the game of Go. AlphaGo was not explicitly programmed to play Go. It was fed millions of games of Go, and developed its own model of the game and strategies of play.

“Neural network” as used herein means a computational model, architecture, or system made up of a number of simple, highly interconnected processing elements which process information by their dynamic state response to external inputs, and is thus able to “learn” information by recognizing patterns or trends. Neural networks, also sometimes known as “artificial neural networks” are based on our understanding of the structure and functions of biological neural networks, such as the brains of mammals. A neural network is a framework for application of machine learning algorithms.

“Static noise image”, also referred to as a “static image” or “noise image” as used herein means an image with random or pseudo-random content. In some embodiments, the random or pseudo-random content will be at the pixel level, or be at the level of groups, areas, or regions of pixels. In some embodiments, the random or pseudo-random content may comprise a single bit per pixel, representing either black or white for that pixel. In some embodiments, the randomness property of the image may be the 2D maximum information entropy, or Shannon Entropy, as part of its properties. Shannon Entropy refers to the amount of randomness or possible information a given statistical model can have, expressed in bits. For example, a coin flip has an information entropy or Shannon Entropy of 1, while m coin flips have an entropy value of m , because a single coin flip has a value of 0 (tails) or 1 (heads), represented by 1 bit. The higher the entropy value for a given object, the more possible states it may be in, which makes it harder to predict the state of, but more malleable for transforming into many possible states and acquire an estimation of another state, for example, transforming into an approximation of another image. In some embodiments, the random or pseudo-random content may further comprise grayscale or color information. In some embodiments, the random or pseudo-random content may comprise vector graphics or other forms of image representation instead of bitmaps or pixels.

“Video” as used herein means a digital representation of a sequence of images representing movement. This digital representation may be in any video file format, of which there already exist a great number. Video files typically contain coded video (visual) data and audio (audible) data in a “container.” This application is primarily concerned with the video data portion of a video file. For exemplary purposes, a non-exhaustive and non-limiting list of video formats includes Audio Video Interleave (avi), Motion Pictures Expert Group (mpg, mpeg, mp2, mp4, etc.), Apple video format (m4v), and Windows Media Video (wmv).

Conceptual Architecture

FIG. 1 (PRIOR ART) is a method diagram illustrating the HEVC video compression and the similar BPG image compression methodology. First, an image or video file is input into a compression codec or compression engine 110, before data files are encoded with intra-frame chunks 120 which remain unchanging through frames of a video in the case of HEVC video file compression, meaning that between frames of a video where a grouping of pixels do not change or change very little, that grouping is not re-encoded in multiple frames of the video 130, thereby reducing the size of the video with only a minor loss of movement in frames. In the case of BPG formatting for still images rather than video for HEVC video formatting, clusters of similar data

including color depth are maintained for groups of pixels in a still image 130, representing a decrease in the data required for equivalently sized images, resulting in a compressed format using similar techniques for both video and still images 140.

FIG. 2 is a diagram illustrating an exemplary algorithm for implementation of an aspect of a preferred embodiment. In this algorithm, a desired image 201 and a noise image 203 are input into a function f 202. Function f 202 is a neural network configured to map noise image 203 to a desired image 201. The output of function f 202 is a set of hyperparameters P 204, or “weights” of function f 202, that map the noise image 202 to the desired or target image 201 within a specified margin of error. The mathematical formula for this algorithm is:

Given: I, N , find θ , such that:

$$f: R^{W_N \times H_N \times C_N} \rightarrow R^{I_N \times I_N \times I_N}, \text{ where } f(\theta|N) = I' \text{ and } h(I', I) \leq$$

The algorithm may be generally described as follows.

Given a color image I (which corresponds to target or desired image 201), with width W_I , height H_I and depth C_I , there is an algorithm 220 which may find a mapping between a given max-entropy image N of size $W_N \times H_N \times C_N$ and I , by fine-tuning the hyperparameters θ of a known function f 202. However, such an algorithm is only capable of reconstructing an approximation of the original image, allowing an error up to a defined threshold. The algorithm 220 measures the error between two images through a function h which may be one of several possible image comparison functions, such as a pixel-wise mean-square-error function. If for all images I there exists an f 202 with θ_f such that the above statement holds true, then it is possible to approximately represent all images via a triplet of $\{f, \theta_f, N\}$. If function f 202 is fixed, and its input N , the only thing missing to create I' is θ . If the file size of θ is smaller than the file size of I , the algorithm has successfully lossy-compressed I , because one only need send θ with a bitstream of size $\text{filesize}(\theta)$ to create I' from the triplet $\{f, \theta, N\}$ at a destination device, by inputting noise image 206 and parameters θ 204 into f 205 on the destination/receiving side to obtain approximated target image I' 207.

However, it is not immediately obvious how to choose a function f that will efficiently find θ . The solution is to use a neural network with its typical structure of convolution \rightarrow bias \rightarrow non-linearity \rightarrow repeat as function f . By using a neural network as function f , the solution to finding its hyperparameters θ given I becomes a machine learning training problem that can be solved using, for example, stochastic gradient descent (SGD). In short, SGD is used to learn the hyperparameters (aka weights) which minimize the error function h , thus obtaining θ for I .

After hyperparameters θ 204 are output from a neural network 202, they may be input into a separate instance of the same or a similar neural network 205, along with a separate instance of an identical noise image 206, to produce an approximation of a desired image I 207. In this way, the system operates as compressor and decompressor, by generating one of two objects depending on what is received as input, either hyperparameters θ 204 or an approximation of an image I 207. The instances of these objects, such as the decompression neural network 205 and associated data, may be operating on different computers, for example a first network endpoint possessing a desired image I 201, function f 202, and noise image 203, may operate on one computer such as a smartphone, laptop, or other computing device, while another function f 205, noise image 206, may exist on another computing device, possibly but not necessarily over

a network. It is possible, for instance, for files to be transferred via a portable memory device, rather than over a network between two computers. It is important to note that the different instances of the neural network at either end of a transmission need not be identical. Neural networks that function similarly will produce an acceptable solution, even if their architecture, programming, or other characteristics are different.

FIG. 3 is an exemplary overall system diagram, according to a preferred embodiment. A network endpoint 301, which may be a laptop or desktop computer, a mobile phone, a workstation, or some other manner of network-enabled computing device, may be connected over a network 306 such as the Internet, and possess a desired image I 304 and noise image N 305. On a network endpoint 301, both images may be used as inputs for an image compression engine 303, which is directly accessible by a network endpoint 301. A desired image I 304 may be in one of many image formats, including .JPG/.JPEG, .PNG, .BMP, or other formats, the specific format of it being a non-crucial element to the instant invention, as new formats may be identified and accounted for in the future as applicable. A static noise image N 305 may similarly be in one of many formats, and may be a singular, unchanging image, which is used in all implementations of the system, or it may be one of several possible images, and sent to the system as input from another source, whether a network endpoint 307 or some other source containing the image. In some embodiments, the static noise image N 305 represents an image that has the highest 2D Shannon Entropy, which may be used for transformation into other images as needed. An image compression engine 303 is an engine present on a network endpoint 301, which may take, as input, a desired image I 304 and static noise image N 305, and execute an image transformation function 202 to generate hyperparameters for the function 202 which, when applied to the noise image N 350, allow the recreation of a close approximation I' (read as "I-prime" and denoting an approximation of I) 310 of a desired image I 304. Specifically, a destination endpoint 307 may receive hyperparameters 308 for input into another instance of an image decompression engine 309, also using a static noise image 311 as input, allowing it to produce an approximation 310, I', of a desired image I 304.

FIG. 4 is an exemplary system diagram of an image compression engine 303, according to a preferred embodiment. As shown in FIG. 3, a noise image N 305, and goal image I 304, which respectively represent a maximum entropy noise image and the initial image to be compressed, are input into a processor 430. A processor 430 is a common computing device used to process arithmetic and logical operations, and is commonly known in the art. A processor 430 has two-way communication with at least a single bit of volatile memory 440, commonly known as Random Access Memory ("RAM"), for short-term storage of data to be accessed by a processor 430, the processor altering data in the RAM and using data in the RAM to alter its own processes. This is commonly understood as one of the key interactions in modern computing. A processor 430 also communicates with a neural network 410, which operates as a function f in an algorithm 220, operating as a framework for machine learning algorithms 420 to process incoming data from a noise image 305 and goal image 304. After an input image 304 is re-created approximately, using the algorithm 220 in FIG. 2, the hyperparameters 302 of a neural network 410 are output to the network endpoint 301 for further use, including transmitting to a destination endpoint,

saving on a hard-drive or external memory device, or other common uses for compressed images.

FIG. 5 is a system diagram of another exemplary embodiment of an image compression engine 500, utilizing a 2D convolution Application Specific Integrated Circuit ("ASIC") 560, according to an embodiment. In this embodiment, a 2D convolution ASIC 560 is utilized to reduce the amount of processing time required for the system to fully compress an image. As shown in FIG. 3, a noise image N 305, and goal image I 304, which respectively represent a maximum entropy noise image and the initial image to be compressed, are input into a processor 530. A processor 530 is a common computing device used to process arithmetic and logical operations, and is commonly known in the art. A processor 530 has two-way communication with at least a single bit of volatile memory 540, commonly known as Random Access Memory ("RAM"), for short-term storage of data to be accessed by a processor 530, the processor altering data in the RAM and using data in the RAM to alter its own processes. This is commonly understood as one of the key interactions in modern computing. A 2D convolution ASIC 511 also communicates bi-directionally with both the system processor 530 and memory 540, in order to handle specialized processing of convolutional neural network 510 data much faster than a general central processing unit might otherwise achieve. A processor 530 also communicates with a neural network 510, which operates as a function f in an algorithm 220, operating as a framework for machine learning algorithms 520 to process incoming data from a noise image 305 and goal image 304. After an input image 304 is re-created approximately, using the algorithm 220 in FIG. 2, the hyperparameters 302 of a neural network 510 are output to the network endpoint 301 for further use, including transmitting to a destination endpoint, saving on a hard-drive or external memory device, or other common uses for compressed images.

FIG. 6 is a flowchart illustrating the processing and compression of data through the system, according to an aspect. A desired image I may be input into the system 601, this being the image that the system will attempt to transform a noise image into with specific hyperparameters. A check for a noise image is performed 602, which may yield either positive or negative results, regarding whether the system already possesses a static noise image 305. If there is no static noise image currently present in the system, a noise image may be input into the system 603, through a network endpoint 307 which may possess a static noise image, or it may be generated automatically, or come from some other source. If a static noise image is already present, or after it is received or input into the system 603, the system is at a state of both the static noise image N 305 and desired image I 304 being present in the system. It is at this point that an image transformation function f may be utilized or executed 604 to apply to a static noise image N 305, the result of the transformation being checked for closeness to a desired image I 605. the closeness being defined by an error value, the hyperparameters of the neural network are output 607, and the system ends execution, ready to receive a new input image and begin execution again 601. If the image transformation function f 410 operating on a noise image N 305 does not yield a sufficiently close result to desired image I 304, then a neural network 410 may alter the alterable hyperparameters θ 302 of a transformation neural network f 410, 606 to attempt to produce ever-closer results, until the output is sufficiently close to an image I 304, 605.

FIG. 7 is a flowchart illustrating the processing and de-compression of data through the system, according to an

aspect. First, hyperparameters are received and input into an image compression engine 701, which apply to the neural network 410 to determine the image transformation and convolution of a noise image into a desired image. The system is checked to ensure that a static noise image N is present 702, and if no noise image is present, such an image is input into the system 703, which may be sent from the same source as the alterable hyperparameters θ , or may be inserted by a separate system, or manually input by a user on the system itself. After a noise image N is inserted into the system 703, or if the noise image is already present, for example if the noise image remains static and identical across all implementations of the disclosed invention, then an image transformation function f may be executed 704, however instead of having no input hyperparameters as in FIG. 6, the hyperparameters are already input into the system 701, and the static noise image 305 is transformed 704 into an image I', which is an approximation of a desired image I 304, before being output 705. In this way, image compression is performed by specifying how to re-create an image rather than altering the image data itself, allowing the specifications for re-creating the image to be transmitted rather than the image itself, ensuring excellent compression of data.

FIG. 8 is a method diagram of high-level components used in the operation of a system for lossy image compression utilizing neural networks, according to a preferred embodiment. First, a desired image must either be present, or input 801, this image being the image that will be attempted to be re-created with an image transformation network 410, before a static noise image N 305 is accessed 802. This noise image will be utilized by an image transformation function 410 to attempt to translate it, using alterable hyperparameters, into an approximation of an input desired image I 304. After a noise image is received or accessed if it is static and unchanging 304, the image transformation function f 410 is executed 803, utilizing alterable hyperparameters θ 302 and feeding into a set of machine learning algorithms 250 operating on a neural network 260, resulting in alterable hyperparameters θ for image transformation function f being altered 804, 805 in an attempt to translate a closer approximation to input image I 304. When an image is produced from a transformation of a noise image N 305, the alterable hyperparameters θ 302 are output 806, such that a user with the image transformation function 410, noise image 305, and hyperparameters 302, may re-create a close approximation of the original desired image 304. In this way, the image may be heavily compressed, and potentially encrypted from users without access to the image transformation function itself. Alterable hyperparameters θ 302 may then be sent to another network endpoint 307, 807, such as another laptop, desktop, or workstation device, or other device capable of running a 2D convolution ASIC as specified in FIG. 5. A destination network endpoint 307 may then, using alterable hyperparameters θ 302, execute an image transformation function f 808, inputting alterable hyperparameters θ 809 as the parameters in which a neural network 410 transforms a noise image 305 to an approximate of a desired image I 304, 810.

FIG. 9 is a state diagram of two users, one encoding and one decoding an image, using the disclosed compression system. In a first state 910, a user A has a desired image I 304, while a user B wants to receive an image I 304 and does not yet possess it. This is analogous to a user possessing an image to compress and send to another user, perhaps a friend, work colleague, or third-party service which requires for one of many possible reasons that the file be compressed

first. In a second state 920, which necessarily follows from the first, training is undergone with a neural network 410 which allows for the hyperparameters θ 302 to be tuned to allow for an image transformation function to transform a static noise image N 305 into an approximation of a desired image I 304. A third state 930, proceeding from the second, allows for user A to acquire the parameters θ 302 as mentioned previously, from the training, before proceeding to a fourth state 940, in which hyperparameters θ 302 are sent from user A to user B, which allows for a fifth state to be reached 950, whereby user B, now possessing the hyperparameters θ 302, may acquire I 304 by utilizing an image transformation function f 410 to transform a noise image N 305 into an approximation of a desired image I 304.

FIG. 10 is a method diagram illustrating training a neural network with a single image, as opposed to a plurality of images to learn from, resulting in a non-general approach to image compression. A desired image is input into the system 1010, as has been elaborated on previously in the disclosed system, such an image comprising any two-dimensional graphical file of one of the numerous viable formats including .BMP, .PNG, or others. A neural network 410 is trained 1020 using on a static noise image N 305, without changing or swapping out the image 305, resulting in a network trained on a single datapoint rather than a large dataset, unlike generalized neural networks. Rather than feeding a large dataset to a network, edge weights of a network are adjusted based on how close a given output of a network's image transformation has come to a desired image I 304, 1030. In this way, training is continued on a network 410 on a single noise image 305, using adjusted weights and parameters 1040. A table 1050 illustrates the differences, and relationship, between a traditional neural network which reaches generalized solutions to problems, and the proposed neural network 410, which utilizes a specialized method of training and does not learn general image transformation techniques, by training on a single noise image iteratively, altering the weights and parameters of the network as it generates new transformed images until a close approximation to the desired image is generated.

FIG. 11 is a diagram illustrating an actual image, and a noise image, going into an image compression engine, parameters being identified which allow for a noise image to be approximately translated to an initial non-noise image, and relayed to a second system over a network to re-generate an approximation of the original image from a noise image. A sample image 1101, and a static noise image 1102, are input into an image compression engine 303. Hyperparameters 303 for re-creating image 1101 are devised through training a neural network to transform a noise image 1102 into a close approximation of a desired image 1101, the transformation being possible due to the noise image 1102 being a maximally entropic image capable of being transformed into any number of images with the correct transformation steps. Hyperparameters 302 for re-creating an image 1101 are sent over a network 306, where they are fed into an image decompression engine 309, along with a copy 1103 of the static noise image 1102, which outputs a close approximation 1104 of the desired image 1101. Note that this is not an identical copy, but a close approximation, therefore illustrating a lossy compression method, and creating a separate image file 1104, not identical to the original image 1101.

FIG. 12 is a system diagram of an image decompression engine 1200 used for decompressing an image, according to an embodiment. It should be noted that the image decompression engine 1200 uses the same components as the

image compression engine 303, but is configured to receive a noise image 305 and hyperparameters 302, and to output an approximated image I' 310. As shown in the right-hand portion of FIG. 12, a noise image N 305, and alterable hyperparameters 302 of a neural network 1210, are input into a processor 1230. A processor 1230 is a common computing device used to process arithmetic and logical operations, and is commonly known in the art. A processor 1230 has two-way communication with at least a single bit of volatile memory 1240, commonly known as Random Access Memory ("RAM"), for short-term storage of data to be accessed by a processor 1230, the processor altering data in the RAM and using data in the RAM to alter its own processes. This is commonly understood as one of the key interactions in modern computing. A processor 1230 also communicates with a neural network 1210, which operates as a function f in an algorithm 220, operating as a framework for machine learning algorithms 1220 to process incoming data from a noise image 305 and, for the purposes of de-compression, the hyperparameters θ 302 of a previous neural network's compression of a goal image 304. After hyperparameters θ 302 are input into a processor 1230, the processor directs the neural network 1210 to incorporate and use these hyperparameters, directing the operation of the network as it transforms a noise image 305 into an approximation 310 of an input image 304, using the algorithm 220 in FIG. 2, the approximation image I' 310 being output to the network endpoint 307 for further use, including transmitting to a destination endpoint, saving on a hard-drive or external memory device, or other common uses for now uncompressed images.

FIG. 13 is a system diagram of an alternative embodiment of an image decompression engine 1310 arrangement being used for decompression while utilizing a specialized 2D convolution Application Specific Integrated Circuit ("ASIC") 1311, according to an embodiment. It should be noted that the image decompression engine 1300 uses the same components as the image compression engine 309, but is configured to receive a noise image 305 and hyperparameters 302, and to output an approximated image I' 310. As shown in the right-hand portion of FIG. 3, a noise image N 305, and alterable hyperparameters 302 of a neural network 1310, are input into a processor 1330. A processor 1330 is a common computing device used to process arithmetic and logical operations, and is commonly known in the art. A processor 1330 has two-way communication with at least a single bit of volatile memory 1340, commonly known as Random Access Memory ("RAM"), for short-term storage of data to be accessed by a processor 1330, the processor altering data in the RAM and using data in the RAM to alter its own processes. This is commonly understood as one of the key interactions in modern computing. A 2D convolution ASIC 1311 also communicates bi-directionally with both the system processor 1330 and memory 1340, in order to handle specialized processing of convolutional neural network 1310 data much faster than a general central processing unit might otherwise achieve. A processor 1330 also communicates with a neural network 1310, which operates as a function f in an algorithm 220, operating as a framework for machine learning algorithms 1320 to process incoming data from a noise image 305 and, for the purposes of de-compression, the hyperparameters θ 302 of a previous neural network's compression of a goal image 304. After hyperparameters θ 302 are input into a processor 1330, the processor directs the neural network 1310 to incorporate and use these hyperparameters, directing the operation of the network as it transforms a noise image 305 into an approxi-

mation 310 of an input image 304, using the algorithm 220 in FIG. 2, the approximation image I' 310 being output to the network endpoint 307 for further use, including transmitting to a destination endpoint, saving on a hard-drive or external memory device, or other common uses for now uncompressed images.

Because training of a single neural network to effectively map a known noise image to a desired image typically takes a highly-optimized graphical processing unit (GPU)-based machine many hours to accomplish for each desired image to be compressed, using normal training methods based on (for example) steepest gradient descent, it is crucial to find a different way to efficiently determine the weights of f for a given desired image. According to an aspect, the inventors have determined that it is possible to generate weights for f , to arbitrary accuracy (in the sense of how closely an output image obtained by applying N to f matches the input image used to generate the respective weights of f), in a single pass through a metanetwork described below. Specifically, by passing the desired image I and the known noise image N as inputs to a function g using a metanetwork, the weights for f are obtained in a single pass, as described below with reference to FIGS. 14-17. More precisely, the inventors have shown that a metanetwork arranged as shown in FIG. 17 demonstrated a massively increased encoding efficiency with regards to runtime compression of images. Specifically, training via SGD on high-end GPU's took more than 6 hours. In contrast to that, the runtime of g is less than a second on similar hardware, and can be less than 100 milliseconds if specialized hardware (such as hardware-based optimized convolution stages).

FIG. 14 is a system diagram of high-level components used in the operation of a system using a metanetwork to achieve image or video compression, and decompression, according to a preferred embodiment. As shown on the left of FIG. 14, a given goal image I 304 is provided to a metanetwork engine 1410, alongside a static (that is, known and unchanging) noise image N 305. Metanetwork engine 1410 comprises a "network of neural networks", including multiple single neural networks that each perform processing on a portion of the goal image I 304, such as specific features or color spectra (for example, in a three-network metanetwork, each network may focus on one of each of the red, blue, and green color bands within the image). Metanetwork engine 1410 uses convolutional training between the single networks within it to arrive at the set of hyperparameters 302 that are appropriate for any given image decompression engine 309 to reconstruct the goal image 304 (or an arbitrarily-similar recreation thereof) from the noise image 305. This set of hyperparameters 302 may then be stored for repeated future use, for example to enable many clients to reconstruct the goal image 304 from a single set of calculated hyperparameters (for example, as may be useful for a photo or video streaming service, wherein the same content is sent repeatedly to a large number of destinations), and may be sent via a network 306 from one endpoint 301 to another 307. At the network destination, the received set of hyperparameters 308 are then provided to an image decompression engine 309 as input along with the noise image I 311, so that the image decompression engine 309 may produce a reasonably-close recreation of the goal image, the recreated image being denoted as I' 310.

FIG. 15 is a system overview diagram of a metanetwork engine 1410, used to train a metanetwork with a set of training images, and used to compress and find filters that may be used to transform a noise image into an approximation of an input image, according to a preferred embodiment.

According to the embodiment, a given goal image I 304 is provided to a metanetwork engine 1410, alongside a static noise image N 305 as inputs. Metanetwork engine 1410 comprises a processor 1540 and memory 1559 for performing the machine learning tasks associated with the training and transformation processes. A neural network 1510 trains on a set of training images 1530 using machine learning algorithms 1520 to arrive at an appropriate set of hyperparameters for operation, the hyperparameters comprising the set of hyperparameters necessary for the metanetwork engine 1410 to create a set of alterable hyperparameters 302 to store and transmit to target image encoding networks, given the inputs of N 305 and I 304. In other words, the set of hyperparameters determined through training is the set used for operation of the metanetwork, and said operation (when given the correct, trained set of hyperparameters) then produces the desired set of hyperparameters 302 from the noise 305 and target 304 images.

FIG. 16 is a dataflow diagram of a system 1610 for lossy compression utilizing a metanetwork 1602 to train, compress, and send data for decompression to a system utilizing a specific neural network configuration 1640, according to an embodiment. According to the embodiment, metanetwork 1602 instantiates a function g , wherein g , when given a known noise image 1603 and an original image 1601 as inputs, determines a specific set of hyperparameters 1604 for the given original image I 1601 and noise image N 1603 as inputs. The hyperparameters 1604 have the property that, when inserted (for example, at a destination device) as weights into a neural network to define a specific instance of a decompression function f 1640, which can be used to map the same known/static noise image 1630 (here, “the same” means noise image 1630 is the same image as noise image 1603 used as input to g in metanetwork 1602 to produce hyperparameters 1604. According to a preferred aspect, hyperparameters 1604 may be transmitted to a destination image encoding network 1640, which comprises a function f such that (produces a facsimile of the original image I, dubbed I' 1650, given the noise image N 1630 and hyperparameters 1604 as inputs. Stated mathematically, this operation takes the form of the set of functions 1620:

Given: $\{I_1, \dots, I_K\}, N, \epsilon, f$

Find: g

So that for $j=\{1, \dots, K\}$: $g(I_j, N)=\theta^j$

$f(g(I_j, N)|N)=I_j$

$h(I'_j, I_j) \leq \epsilon \Leftrightarrow h(f(g(I_j, N)|N), I_j) \leq \epsilon$

FIG. 17 is a system diagram of multiple individual neural networks 1710a communicating with each other within a metanetwork 1710 to produce a plurality of sets of filter weights 1721, 1723, 1725, which may be inserted into neural networks 1722, 1724, and 1726, which collectively embody a specific instance of function f that may be used to map noise image 1701 to image 1730, which is close—to within an arbitrary error limit—to identical to input image 1711. As shown, a metanetwork 1710 is a network of neural networks, comprising a plurality of individual networks that each perform independent machine learning tasks focused on a portion of an original image 1711. As shown, one arrangement uses three individual networks (and the inventor has determined through practice that this configuration provides excellent performance for the relative amount of resources utilized, reducing an image compression task from hours to milliseconds), however alternate metanetwork arrangements are possible according to the embodiment such as using only two individual networks, or using four or more individual networks.

When an original image 1711 is provided as an input to an individual network 1710a (while only this specific individual network is highlighted within the figure, this was done for the sake of clarity and it should be appreciated that the discussion of operation of 1710a applies to any and all individual networks present in the metanetwork 1710), a sequence of convolutional filters 1714a-n are applied, wherein the output of each filter may be provided as an input to the next, to enable machine learning. Each individual network's convolutional outputs may also be provided as inputs to the next sequential filter in another individual network's processing 1712a-n, 1713a-n, so that the networks effectively learn from each other's processing and collectively “zero in” on ideal solutions. After this convolutional processing, a number of non-convolutional final filters 1715a-n, 1716a-n, 1717a-n may be applied to each individual network's output (and in this case, only that network's output), using the static noise image 1701 as an added input.

This produces a set of filters 1721, 1723, 1725 that are based on the combined development of the convolutional, collective processing of the original image, as well as the network-specific processing based on the noise image, such that each filter is different (being based on only a single network's processing, and therefore varying from one another due to variances in the networks and their respective outputs and learning processes), but the combination of filters, when applied in sequence at a target image encoding network 1720, reconstructs successively more-accurate representations 1722, 1724, 1726 before arriving at a final, arbitrarily-similar reconstruction 1730 of the original image 1711.

It will be seen, in FIG. 17, that data is passed between layers within a given convolutional neural network (e.g., 1712a-n), as is conventional. But additionally, inter-metanetwork connections are made, according to an aspect, linking outputs of one neural network (one horizontal row) of a metanetwork (for example, 1712a-n) to the input of a next stage of a different neural network (a different horizontal row, such as 1713a-n), as shown in FIG. 17. During some initial tests of multiple metanetworks without these inter-metanetwork connections, training and quality challenges were encountered. The training challenge was the vanishing gradient problem. This occurs when the updates the learning algorithm wants to make to the weights become extremely small (in the order of 10^{-8}) due to continuous multiplications by small values. These small updates result in a network which becomes “stale” and stops learning; in other words, the function stops improving. As done in intra-metanetwork connections, this problem may be fixed by adding skip connections in between the layers within a metanetwork to shorten the route an update has to take to get to its parameter, lowering the order by which the update diminishes. This technique was fully utilized in an exemplary aspect, with substantial improvements noted.

To further improve the flow of information, inter-metanetwork pathways are added. These act as information-injections between the metanetworks. Their main role is to update the current metanetwork on the state of the previous ones. This further helped the flow of gradient information during training, leading to a training process that is more stable while requiring less time.

Another reason for adding inter-metanetwork pathways is so the current metanetwork may be aware of the filter (and therefore transformation) the previous network produced, and it may produce its filter in a way that will complement the transformation of the previous filter rather than cancel it

out. This ability to know the state of the previous metanetwork is important since in image encoding networks according to aspects of the invention, the transformations to the input (noise) are typically applied in order. Therefore, if a metanetwork is tasked with predicting a transformation at a particular stage, it should know the previous transformations that have already been applied, in other words, the state of the previous metanetwork rows.

Each metanetwork is tasked with predicting the filter which will most adequately transform an image-encoding network input to a step closer to a desired image. The input to the metanetwork is just the desired image (the known noise image is injected late in the first row at layers 1715a-n, as shown in FIG. 17), so the metanetwork is seeing an incomplete picture. It does not see the “from” image but just the “to” image. Hence, a late fusion of the IEN input (noise) is provided to the metanetwork. This increases the information the metanetwork can use to make its prediction, and gives a boost in image quality. Furthermore, this fusion of information is done in an “or” manner, so the metanetwork can choose if it requires it or not.

To summarize, a method according to an aspect, such as that shown in FIG. 17, may be viewed as an ensemble of metanetworks, since the method uses a collection or group of metanetworks. One might argue that it’s the same network; however, this is false since all individual metanetworks are independent from one another with regards to their trainable parameters, inputs and outputs. It is not even necessary to train them together, but it is convenient.

Metanetwork 1710 must be trained before it can carry out its function of determining the weights 1722, 1723, 1725 required for function f to map noise image N 1701 to target image 1730. Accordingly, FIG. 18 is a method diagram illustrating steps required for training metanetwork 1710 for use in lossy compression of images and video using a metanetwork. In an initial step 1810, a set of training images j are provided to a metanetwork as inputs. These are used to train the metanetwork based on the images j and a given noise image N 1820. This training, as described above in FIG. 15, arrives at a set of filters needed to apply to the noise image N to arrive at a reconstruction of a given original image I 1830. In more detail, this reconstruction process involves a number of individual networks within the metanetwork, each of which generates a subset of hyperparameters 1840, and each of which subset is different from the others and focused on a specific portion or attribute of the input image I 1850. Each subsequent individual network takes, as input, the states of all previous networks 1860 so that their collective machine learning is used as a convolutional neural network built out of smaller, more specialized convolutional neural networks (each being focused on a specific attribute of the image, while the overall metanetwork being focused on the whole by using each specific network as a convolutional filter within its own processing). To reconstruct the target image, static image N is passed through the successive filters developed by the networks 1870, producing a close approximation image I' 1880. When the approximation image is determined to be acceptable, that set of hyperparameters is stored and training is considered complete; the hyperparameters may then be sent 1890 to any arbitrary number of destination image encoding networks, each of which may then use the hyperparameters and noise image N to rapidly produce the reconstructed image I'.

FIG. 19 is a flowchart of the steps taken for a single network within a metanetwork to train on a set of images and produce a network operating as a portion of function g, for neural network hyperparameter prediction of an image

encoding network f, according to an embodiment. In an initial step 1901, a set of training images j is input to the metanetwork. At each individual network within the metanetwork, a check is performed 1902 to determine if a noise image is present. If no noise image is present, one is provided 1903, and then (or if the noise image was already present) the metanetwork executes the image transformation function g on each of the training images j_n 1904. After performing the transformation, the result is checked to determine whether the output image is acceptably close to a desired image I 1905, and if not then each individual network within the metanetwork alters its parameters slightly 1906 and repeats the function g, to iterate closer to the desired output result. When the recreated image I' is acceptably close to the original I, the set of hyperparameters is stored as the acceptable set of hyperparameters needed for an acceptable image transformation 1907.

FIG. 20 is a flowchart of the process of multiple networks communicating within a metanetwork for the purposes of cross-training and developing progressive filters to transform a static image with, to help alleviate the vanishing gradient problem, according to an embodiment. In an initial step 2010, a convolutional processing layer i is executed across all single networks within a metanetwork. Then 2020, a check is performed to determine if additional layers exist, waiting to be performed. If so, then each individual network sends, as output, the results of its respective processing layer i to be used as an input for all next networks' next convolutional layer, i+1 2040. Then, i increments 2050 and operation continues 2010. When no more layers exist, convolutional processing concludes and each individual network then produces its respective filter 2030, thereby producing the complete set of filters needed to recreate a target image from a noise image.

Hardware Architecture

Generally, the techniques disclosed herein may be implemented on hardware or a combination of software and hardware. For example, they may be implemented in an operating system kernel, in a separate user process, in a library package bound into network applications, on a specially constructed machine, on an application-specific integrated circuit (“ASIC”), or on a network interface card.

Software/hardware hybrid implementations of at least some of the aspects disclosed herein may be implemented on a programmable network-resident machine (which should be understood to include intermittently connected network-aware machines) selectively activated or reconfigured by a computer program stored in memory. Such network devices may have multiple network interfaces that may be configured or designed to utilize different types of network communication protocols. A general architecture for some of these machines may be described herein in order to illustrate one or more exemplary means by which a given unit of functionality may be implemented. According to specific aspects, at least some of the features or functionalities of the various aspects disclosed herein may be implemented on one or more general-purpose computers associated with one or more networks, such as for example an end-user computer system, a client computer, a network server or other server system, a mobile computing device (e.g., tablet computing device, mobile phone, smartphone, laptop, or other appropriate computing device), a consumer electronic device, a music player, or any other suitable electronic device, router, switch, or other suitable device, or any combination thereof. In at least some aspects, at least some of the features or functionalities of the various aspects disclosed herein may be implemented in one or more virtualized computing

environments (e.g., network computing clouds, virtual machines hosted on one or more physical computing machines, or other appropriate virtual environments).

Referring now to FIG. 21, there is shown a block diagram depicting an exemplary computing device **10** suitable for implementing at least a portion of the features or functionalities disclosed herein. Computing device **10** may be, for example, any one of the computing machines listed in the previous paragraph, or indeed any other electronic device capable of executing software- or hardware-based instructions according to one or more programs stored in memory. Computing device **10** may be configured to communicate with a plurality of other computing devices, such as clients or servers, over communications networks such as a wide area network, a metropolitan area network, a local area network, a wireless network, the Internet, or any other network, using known protocols for such communication, whether wireless or wired.

In one embodiment, computing device **10** includes one or more central processing units (CPU) **12**, one or more interfaces **15**, and one or more busses **14** (such as a peripheral component interconnect (PCI) bus). When acting under the control of appropriate software or firmware, CPU **12** may be responsible for implementing specific functions associated with the functions of a specifically configured computing device or machine. For example, in at least one embodiment, a computing device **10** may be configured or designed to function as a server system utilizing CPU **12**, local memory **11** and/or remote memory **16**, and interface(s) **15**. In at least one embodiment, CPU **12** may be caused to perform one or more of the different types of functions and/or operations under the control of software modules or components, which for example, may include an operating system and any appropriate applications software, drivers, and the like.

CPU **12** may include one or more processors **13** such as, for example, a processor from one of the Intel, ARM, Qualcomm, and AMD families of microprocessors. In some embodiments, processors **13** may include specially designed hardware such as application-specific integrated circuits (ASICs), electrically erasable programmable read-only memories (EEPROMs), field-programmable gate arrays (FPGAs), and so forth, for controlling operations of computing device **10**. In a specific embodiment, a local memory **11** (such as non-volatile random-access memory (RAM) and/or read-only memory (ROM), including for example one or more levels of cached memory) may also form part of CPU **12**. However, there are many different ways in which memory may be coupled to system **10**. Memory **11** may be used for a variety of purposes such as, for example, caching and/or storing data, programming instructions, and the like. It should be further appreciated that CPU **12** may be one of a variety of system-on-a-chip (SOC) type hardware that may include additional hardware such as memory or graphics processing chips, such as a QUALCOMM SNAP-DRAGON™ or SAMSUNG EXYNOS™ CPU as are becoming increasingly common in the art, such as for use in mobile devices or integrated devices.

As used herein, the term “processor” is not limited merely to those integrated circuits referred to in the art as a processor, a mobile processor, or a microprocessor, but broadly refers to a microcontroller, a microcomputer, a programmable logic controller, an application-specific integrated circuit, and any other programmable circuit.

In one embodiment, interfaces **15** are provided as network interface cards (NICs). Generally, NICs control the sending and receiving of data packets over a computer network; other types of interfaces **15** may for example support other

peripherals used with computing device **10**. Among the interfaces that may be provided are Ethernet interfaces, frame relay interfaces, cable interfaces, DSL interfaces, token ring interfaces, graphics interfaces, and the like. In addition, various types of interfaces may be provided such as, for example, universal serial bus (USB), Serial, Ethernet, FIREWIRE™, THUNDERBOLT™, PCI, parallel, radio frequency (RF), BLUETOOTH™, near-field communications (e.g., using near-field magnetics), 802.11 (WiFi), frame relay, TCP/IP, ISDN, fast Ethernet interfaces, Gigabit Ethernet interfaces, Serial ATA (SATA) or external SATA (ESATA) interfaces, high-definition multimedia interface (HDMI), digital visual interface (DVI), analog or digital audio interfaces, asynchronous transfer mode (ATM) interfaces, high-speed serial interface (HSSI) interfaces, Point of Sale (POS) interfaces, fiber data distributed interfaces (FDDIs), and the like. Generally, such interfaces **15** may include physical ports appropriate for communication with appropriate media. In some cases, they may also include an independent processor (such as a dedicated audio or video processor, as is common in the art for high-fidelity A/V hardware interfaces) and, in some instances, volatile and/or non-volatile memory (e.g., RAM).

Although the system shown in FIG. 21 illustrates one specific architecture for a computing device **10** for implementing one or more of the inventions described herein, it is by no means the only device architecture on which at least a portion of the features and techniques described herein may be implemented. For example, architectures having one or any number of processors **13** may be used, and such processors **13** may be present in a single device or distributed among any number of devices. In one embodiment, a single processor **13** handles communications as well as routing computations, while in other embodiments a separate dedicated communications processor may be provided. In various embodiments, different types of features or functionalities may be implemented in a system according to the invention that includes a client device (such as a tablet device or smartphone running client software) and server systems (such as a server system described in more detail below).

Regardless of network device configuration, the system of the present invention may employ one or more memories or memory modules (such as, for example, remote memory block **16** and local memory **11**) configured to store data, program instructions for the general-purpose network operations, or other information relating to the functionality of the embodiments described herein (or any combinations of the above). Program instructions may control execution of or comprise an operating system and/or one or more applications, for example. Memory **16** or memories **11**, **16** may also be configured to store data structures, configuration data, encryption data, historical system operations information, or any other specific or generic non-program information described herein.

Because such information and program instructions may be employed to implement one or more systems or methods described herein, at least some network device embodiments may include nontransitory machine-readable storage media, which, for example, may be configured or designed to store program instructions, state information, and the like for performing various operations described herein. Examples of such nontransitory machine-readable storage media include, but are not limited to, magnetic media such as hard disks, floppy disks, and magnetic tape; optical media such as CD-ROM disks; magneto-optical media such as optical disks, and hardware devices that are specially configured to

21

store and perform program instructions, such as read-only memory devices (ROM), flash memory (as is common in mobile devices and integrated systems), solid state drives (SSD) and “hybrid SSD” storage drives that may combine physical components of solid state and hard disk drives in a single hardware device (as are becoming increasingly common in the art with regard to personal computers), memristor memory, random access memory (RAM), and the like. It should be appreciated that such storage means may be integral and non-removable (such as RAM hardware modules that may be soldered onto a motherboard or otherwise integrated into an electronic device), or they may be removable such as swappable flash memory modules (such as “thumb drives” or other removable media designed for rapidly exchanging physical storage devices), “hot-swappable” hard disk drives or solid state drives, removable optical storage discs, or other such removable media, and that such integral and removable storage media may be utilized interchangeably. Examples of program instructions include both object code, such as may be produced by a compiler, machine code, such as may be produced by an assembler or a linker, byte code, such as may be generated by for example a JAVA™ compiler and may be executed using a Java virtual machine or equivalent, or files containing higher level code that may be executed by the computer using an interpreter (for example, scripts written in Python, Perl, Ruby, Groovy, or any other scripting language).

In some embodiments, systems according to the present invention may be implemented on a standalone computing system. Referring now to FIG. 22, there is shown a block diagram depicting a typical exemplary architecture of one or more embodiments or components thereof on a standalone computing system. Computing device 20 includes processors 21 that may run software that carry out one or more functions or applications of embodiments of the invention, such as for example a client application 24. Processors 21 may carry out computing instructions under control of an operating system 22 such as, for example, a version of MICROSOFT WINDOWS™ operating system, APPLE OSX™ or iOS™ operating systems, some variety of the Linux operating system, ANDROID™ operating system, or the like. In many cases, one or more shared services 23 may be operable in system 20, and may be useful for providing common services to client applications 24. Services 23 may for example be WINDOWS™ services, user-space common services in a Linux environment, or any other type of common service architecture used with operating system 21. Input devices 28 may be of any type suitable for receiving user input, including for example a keyboard, touchscreen, microphone (for example, for voice input), mouse, touchpad, trackball, or any combination thereof. Output devices 27 may be of any type suitable for providing output to one or more users, whether remote or local to system 20, and may include for example one or more screens for visual output, speakers, printers, or any combination thereof. Memory 25 may be random-access memory having any structure and architecture known in the art, for use by processors 21, for example to run software. Storage devices 26 may be any magnetic, optical, mechanical, memristor, or electrical storage device for storage of data in digital form (such as those described above, referring to FIG. 21). Examples of storage devices 26 include flash memory, magnetic hard drive, CD-ROM, and/or the like.

In some embodiments, systems of the present invention may be implemented on a distributed computing network, such as one having any number of clients and/or servers. Referring now to FIG. 23, there is shown a block diagram

22

depicting an exemplary architecture 30 for implementing at least a portion of a system according to an embodiment of the invention on a distributed computing network. According to the embodiment, any number of clients 33 may be provided. Each client 33 may run software for implementing client-side portions of the present invention; clients may comprise a system 20 such as that illustrated in FIG. 22. In addition, any number of servers 32 may be provided for handling requests received from one or more clients 33. Clients 33 and servers 32 may communicate with one another via one or more electronic networks 31, which may be in various embodiments any of the Internet, a wide area network, a mobile telephony network (such as CDMA or GSM cellular networks), a wireless network (such as WiFi, WiMAX, LTE, and so forth), or a local area network (or indeed any network topology known in the art; the invention does not prefer any one network topology over any other). Networks 31 may be implemented using any known network protocols, including for example wired and/or wireless protocols.

In addition, in some embodiments, servers 32 may call external services 37 when needed to obtain additional information, or to refer to additional data concerning a particular call. Communications with external services 37 may take place, for example, via one or more networks 31. In various embodiments, external services 37 may comprise web-enabled services or functionality related to or installed on the hardware device itself. For example, in an embodiment where client applications 24 are implemented on a smartphone or other electronic device, client applications 24 may obtain information stored in a server system 32 in the cloud or on an external service 37 deployed on one or more of a particular enterprise’s or user’s premises.

In some embodiments of the invention, clients 33 or servers 32 (or both) may make use of one or more specialized services or appliances that may be deployed locally or remotely across one or more networks 31. For example, one or more databases 34 may be used or referred to by one or more embodiments of the invention. It should be understood by one having ordinary skill in the art that databases 34 may be arranged in a wide variety of architectures and using a wide variety of data access and manipulation means. For example, in various embodiments one or more databases 34 may comprise a relational database system using a structured query language (SQL), while others may comprise an alternative data storage technology such as those referred to in the art as “NoSQL” (for example, HADOOP CASSANDRA™, GOOGLE BIGTABLE™, and so forth). In some embodiments, variant database architectures such as column-oriented databases, in-memory databases, clustered databases, distributed databases, or even flat file data repositories may be used according to the invention. It will be appreciated by one having ordinary skill in the art that any combination of known or future database technologies may be used as appropriate, unless a specific database technology or a specific arrangement of components is specified for a particular embodiment herein. Moreover, it should be appreciated that the term “database” as used herein may refer to a physical database machine, a cluster of machines acting as a single database system, or a logical database within an overall database management system. Unless a specific meaning is specified for a given use of the term “database”, it should be construed to mean any of these senses of the word, all of which are understood as a plain meaning of the term “database” by those having ordinary skill in the art.

Similarly, most embodiments of the invention may make use of one or more security systems 36 and configuration

systems **35**. Security and configuration management are common information technology (IT) and web functions, and some amount of each are generally associated with any IT or web systems. It should be understood by one having ordinary skill in the art that any configuration or security subsystems known in the art now or in the future may be used in conjunction with embodiments of the invention without limitation, unless a specific security **36** or configuration system **35** or approach is specifically required by the description of any specific embodiment.

FIG. **24** shows an exemplary overview of a computer system **40** as may be used in any of the various locations throughout the system. It is exemplary of any computer that may execute code to process data. Various modifications and changes may be made to computer system **40** without departing from the broader scope of the system and method disclosed herein. Central processor unit (CPU) **41** is connected to bus **42**, to which bus is also connected memory **43**, nonvolatile memory **44**, display **47**, input/output (I/O) unit **48**, and network interface card (NIC) **53**. I/O unit **48** may, typically, be connected to keyboard **49**, pointing device **50**, hard disk **52**, and real-time clock **51**. NIC **53** connects to network **54**, which may be the Internet or a local network, which local network may or may not have connections to the Internet. Also shown as part of system **40** is power supply unit **45** connected, in this example, to a main alternating current (AC) supply **46**. Not shown are batteries that could be present, and many other devices and modifications that are well known but are not applicable to the specific novel functions of the current system and method disclosed herein. It should be appreciated that some or all components illustrated may be combined, such as in various integrated applications, for example Qualcomm or Samsung system-on-a-chip (SOC) devices, or whenever it may be appropriate to combine multiple capabilities or functions into a single hardware device (for instance, in mobile devices such as smartphones, video game consoles, in-vehicle computer systems such as navigation or multimedia systems in automobiles, or other integrated hardware devices).

In various embodiments, functionality for implementing systems or methods of the present invention may be distributed among any number of client and/or server components. For example, various software modules may be implemented for performing various functions in connection with the present invention, and such modules may be variously implemented to run on server and/or client components.

The skilled person will be aware of a range of possible modifications of the various embodiments described above. Accordingly, the present invention is defined by the claims and their equivalents.

What is claimed is:

1. A system for lossy image and video compression utilizing a metanetwork, comprising:

a metanetwork engine comprising a processor, a memory, and a first plurality of programming instructions stored in the memory, wherein the first plurality of programming instructions, when operating on the processor, cause the processor to:

receive a desired image;

receive a noise image;

receive a set of training images;

using the set of training images, train a plurality of neural networks to reconstruct each of the set of training images by mapping the noise image to each of the set of training images;

store the parameters for each of the plurality of neural networks as a set of metanetwork hyperparameters;

use the set of metanetwork hyperparameters as operating parameters for each of the plurality of neural networks;

use the plurality of neural networks to map the noise image to the desired image, producing a second set of hyperparameters corresponding to the specific filters produced from the operation of each of the plurality of neural networks, such that the second set of hyperparameters, when applied to the noise image using the neural network, produce an approximation of the desired image within an error that is less than a pre-determined threshold; and

store the second set of hyperparameters for use in future image mapping operations.

2. The system of claim **1**, wherein each of the plurality of neural networks:

generates at least one convolutional filter, wherein the noise image may be filtered through all convolutional filters in succession, mapping it to an approximation of a desired image; and

facilitates communication between the plurality of neural networks to alleviate the vanishing gradient problem.

3. The system of claim **2**, wherein the plurality of neural networks may be located on separate computing devices, connected across a network.

4. The system of claim **1**, wherein the noise image is static and unchanging.

5. A method for lossy image compression utilizing a metanetwork, comprising the steps of:

receiving a desired image;

receiving a noise image;

receiving a set of training images;

using the set of training images to train a plurality of neural networks to reconstruct each of the set of training images by mapping the noise image to each of the set of training images;

storing the parameters for each of the plurality of neural networks as a set of metanetwork hyperparameters;

using the set of metanetwork hyperparameters as operating parameters for each of the plurality of neural networks;

using the plurality of neural networks to map the noise image to the desired image, producing a second set of hyperparameters corresponding to the specific filters produced from the operation of each of the plurality of neural networks, such that the second set of hyperparameters, when applied to the noise image using the neural network, produce an approximation of the desired image within an error that is less than a pre-determined threshold; and

storing the second set of hyperparameters for use in future image mapping operations.

6. The method of claim **5**, further comprising the steps of: Generating, at each of the plurality of neural networks, at least one convolutional filter, wherein a noise image may be filtered through the convolutional filters in succession, mapping it to an approximation of a desired image, using a plurality of neural meta-networks; and facilitating communication between the plurality of neural networks to alleviate the vanishing gradient problem.

7. The method of claim **6**, wherein the plurality of neural networks may be located on separate computing devices, connected across a network.

8. The method of claim 5, wherein the noise image is static and unchanging.

* * * * *